Original Article
ISSN (Online): 2582-7472

ShodhKosh: Journal of Visual and Performing Arts
January-June 2026 7(1), 393–405

A PERCEPTUAL HASH-BASED APPROACH TO VISUAL SIMILARITY RECOGNITION IN DIGITAL ART IMAGES

Qiu Yuefu [1], Kazem Chamran [2], Hazirah Bee Yusof [3]

[1] City University Malaysia, 46100 Petaling Jaya, Malaysia
[2] City University Malaysia, 46100 Petaling Jaya, Malaysia
[3] City University Malaysia, 46100 Petaling Jaya, Malaysia

Received 30 January 2026
Accepted 09 March 2026
Published 16 April 2026

Corresponding Author
Qiu Yuefu, cityuqiu@163.com

DOI
10.29121/shodhkosh.v7.i1.2026.7641

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ABSTRACT

With the advancement of artificial intelligence, digital art increasingly depends on image-based creation, distribution, and reproduction through online platforms. This trend has generated significant demand for robust methods to identify visual similarity, safeguard digital artworks, and facilitate intelligent image management. While deep neural network (DNN)-based visual recognition techniques demonstrate strong performance, their substantial size and computational demands often hinder deployment in lightweight application contexts. This study introduces a perceptual hash-based method for visual similarity recognition in digital art images. By combining image preprocessing with a progressive three-tier similarity-matching framework, the approach generates stable, consistent visual fingerprints for images with similar formal attributes. Experimental findings reveal that the proposed method achieves 100% precision and 96% recall, with a payload increase of only 2.47 MB and a memory footprint below 2%. These outcomes suggest that the model is efficient, lightweight, and well-suited for practical applications, including digital artwork authentication, visual archive management, image copyright protection, and intelligent art platform security. This research establishes a technical foundation for integrating computational image analysis with the protection and management of contemporary digital visual culture.

Keywords: Digital Art, Visual Similarity, Perceptual Hash, Image Authentication, Artificial Intelligence, Lightweight Model

1. INTRODUCTION

AI is a driving force behind the intelligent revolution, while blockchain enables automatic, trustworthy transactions between people and digital agents. Together, these technologies form two essential parts of the intelligent revolution. However, both AI and blockchain face security challenges as they develop, and they can also be used to strengthen malware, which plays a major role in cybersecurity Chawande (2024). This includes supporting both attacks and defenses. When used for attacks, this type of malware is known as “intelligent malware” Nadeem et al. (2022). Currently, research on attacking and defending against intelligent malware is a major topic in network security. Throughout the

malware lifecycle, attackers are increasingly using advanced techniques to avoid detection and carry out targeted attacks Shahid (2025).

Referring to the ATT and CK Network Threat framework, the current mainstream environmental awareness technologies include sandbox avoidance technology and environmental keying technology. Sandbox avoidance refers to an attack technology Barchuk and Volkov (2025) that malicious code recognizes sandbox / virtual machine analysis environment and interrupt execution Koch and Begoli (2025). Research for sandbox avoidance has become mature.

Environmental keying technology refers to the malicious code Asghar et al. (2024) based on the target environment information export add/decryption key, using cryptography technology to limit their own execution or operation, the target to achieve accurate identification and strike, to the target attack intention hidden Ding et al. (2024). Riordan and Schneier (1998) introduced the concept of environmental keying technology. The related malicious code is environment-keyed, specifically the IUM. First proposed to use AI technology to achieve a new type of environment keying malicious code DeepLocker, make it have the generalization consistency different from the traditional environmental keying technology. However, the current environmental keying technology is still in the early stage of research. Due to the lack of in-depth research on this kind of attack technology, an effective research system has not been established.

As shown in Figure 1, the AI technology introduced by DeepLocker is represented by its own embedded DNN model. The DNN model is composed of two submodels used for target identification and key generation. DeepLocker During startup, the DNN model starts synchronously, and the target recognition sub-model continuously receives the adapted input image and outputs a high-dimensional feature for each input image; the key generating sub-model continuously receives high-dimensional features and outputs a candidate key for each high-dimensional feature; the candidate key is used to attempt to decrypt the secret load. Among them, the target recognition submodel focuses on accurate recognition ability and generalization ability to a class of things; the key generation submodel focuses on intention hiding ability and consistent realization based on generalization ability. Therefore, on the premise of ensuring that the DNN model is well trained, if the secret load is successfully decrypted, the target face is accurately identified, ensuring the precise attack on the specific target; otherwise, if the secret load is not successfully decrypted, the DeepLocker is located in the non-target environment and ensures the confidentiality of the attack load by hiding the attack intention.

Figure 1

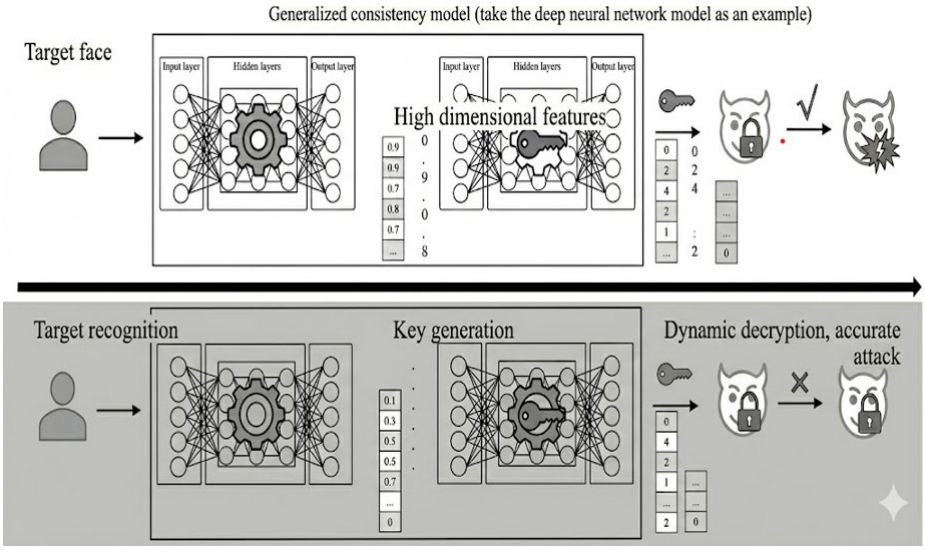

Figure 1 Schematic Representation of the Workflow of the Deeplocker

However, the new attack technology has not been thoroughly analyzed. For example, it is still unclear whether the new environment-keyed malicious code must rely on AI technology or whether other technologies could be used without changing its core function. It is also uncertain whether lightweight alternative technologies are actually feasible. Perceptual hashing is applied to image representations Jain et al. (2023) and involves two main steps: feature extraction and quantization, and compression. Feature extraction extracts the main perceptual features of the image, ensuring robustness to preserve the image content Guo et al. (2025).

Quantization compression is used to ensure the summary of the perceptual hash algorithm. Typically, the extracted features are compressed and encoded into a binary vector, creating a unique 'fingerprint' for the image. There are several types of perceptual hash algorithms, depending on how features are extracted. For example, the average hash algorithm (aHash) Aissi et al. (2024) uses the mean pixel value, the difference hash algorithm (dHash) Dos et al. (2024) uses pixel differences, and the perceptual hash algorithm (pHash) Chen and Cao (2023) uses the image's discrete cosine transform.

Figure 2

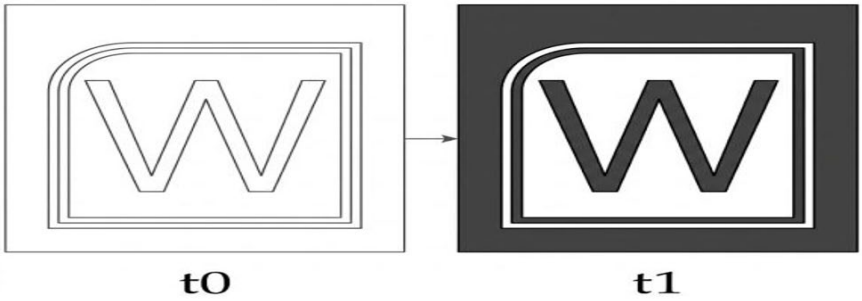

Figure 2 Image with the Same Fingerprint

Figure 2 illustrates that image fingerprint similarity measures facilitate the assessment of similarity between distinct images. A smaller distance value corresponds to greater similarity. Perceptual hash algorithms, therefore, generate identical or similar fingerprints for visually similar image classes. In the context of network security, perceptual hashing demonstrates robust generalization across file variants that exhibit similar behavior. Malware images within the same family tend to be highly similar, while those from different families display significant differences. Consequently, perceptual hashing is widely employed to detect variants of malicious code.

Bhaskara and Bhattacharyya (2018) developed a method for identifying malware features based on perceptual hashing. This approach primarily generates hash signatures for malware program icons and helps extract icon features from malware programs. In the case of malicious repackaged applications, their user interface images closely resemble those of known benign applications. Building on this observation, a dynamic detection method for prepackaged malware was proposed using aware hash techniques. This method achieved a detection accuracy of 88.16% and demonstrated the applicability of aware hash techniques within sandbox-based malware detection. In addition, because the classification model based on AI technology also has the ability to generalize to a class of pictures, some researchers also compared the perception hash with AI technology.

Based on detection accuracy, perceptual hash technology is identified as an effective solution for detecting TOR dark web service domains. Malware variants were visualized as images using GLCM texture feature vectors and a bag model to generate perceptual hashes for variant detection. Experimental results demonstrate that the proposed method achieves higher accuracy and faster detection speed than state-of-the-art deep learning-based detection methods, making it suitable for large-scale detection of malicious software variants. This study also established threat modeling for IUM, focusing on this emerging type of malicious code threat to inform future research. Given the limitations of DeepLocker and leveraging the characteristics of perceptual hash technology, we propose an intelligent malicious code phMalware based on perceptual hash. phMalware through image pre-processing and three layers of similarity matching, accurate target recognition, intention hiding, and generalization consistency ability are realized. Through experimental design, this work compares DeepLocker based on AI technology with phMalware based on perceptual hash model from three levels of measuring generalization consistency, volume increment, and practical feasibility. The comparison results prove the advantages of perceptual hash. phMalware the proposal is a key step towards practice in the laboratory. It not only emphasizes the security threat caused by defense generalization and consistent malicious code, but also makes a fundamental contribution to the research of new malicious code, for instance, pre-defense Nguyen et al. (2020).

Our contributions in this paper are summarized as follows: Construct a systematic threat model for IUM. Explore lightweight technologies to replace massive AI models, proposing a perceptual hash-based intelligent malware. Evaluate its generalization consistency and practical feasibility through systematic comparative experiments. Propose multi-dimensional, forward-looking defense strategies against such novel, highly evasive attacks.

2. METHODS

2.1. INTELLIGENT UNMANNED MALWARE (IUM)

Intelligent Unmanned Malware (IUM) is composed of six tuples, reflecting an executable software that is expected to achieve precise strikes against specific attack targets on the basis of ensuring the core functions of "accurate identification" and "intent concealment". Denoted as IUM =

(iTotalSpace, iKeyGen, iCipher, iKeyDis, iBenCode, iProperty), where iTotalSpace is the input space of IUM, which represents the size of the range of IUM's attempt to find the attack target, and reflects the difficulty of locating the attack target. iKeyGen is the candidate key generator. iCipher denotes the ciphertext attack payload, which carries an uncrackable malicious attack intent. iKeyDis denotes the candidate key authenticator. iBenCode represents the non-cryptographic, executable, benign application. IUM is a benign disguise based on iBenCode. iProperty represents the uncrackable security property that is intrinsic to IUM.

Based on the formal definitions above, this study constructs a seven-stage advanced threat attack chain for IUM: (1) reconnaissance, (2) construction, (3) delivery, (4) Standing/Persistence, (5) Lateral movement, (6) Avoidance, (7) Impact. Attackers first conduct reconnaissance to obtain the unique characteristics of the target host, and use these features to construct highly stealthy decryption triggers via AI or hash black-box techniques. Subsequently, the encrypted attack payload is embedded into a legitimate, benign program for distribution and persistence through supply chain poisoning or lateral movement. Before reaching the target host, IUM behaves as a completely normal program to evade antivirus detection and reverse engineering. Once it arrives at the specific target, it automatically unlocks its intent and executes the attack. Furthermore, because it does not rely on a Command and Control (C2) server, traditional domain blocking and traceback methods are ineffective, posing a severe defensive challenge.

2.2. PERCEPTUAL HASH-BASED INTELLIGENT MALWARE

2.2.1. MODEL ARCHITECTURE

The core of the phMalware construction is the implementation of the generalization consistency model, for which this work designs and implements a generalized consistency perception model (hereinafter referred to as the "perceptual hash model").

The first step, image preprocessing is to convert the input images into standardized images, so that they only focus on the target face images and key facial feature points to lay the foundation for extracting local effective features; the second step, the progressive three-layer similarity matching is divided into σ l similarity matching, σ 2 similarity matching, and key generation. In progressive three-layer similarity matching, σ 1 and σ 2 represent two different similarity measurement thresholds. The key generation stage is the third layer similarity measurement threshold is fixed to 1 to output consistent image fingerprint for a target image to generate a unique encryption and decryption key.

2.2.2. IMAGE PREPROCESSING

This work is based on the open-source tool FaceTools to complete the image preprocessing process. Figure 3 shows the example character image processing process, which consists of three steps: (1) detection, (2) alignment, (3) cropping.

First, to detect the facial contour, accurately identify the position and size of the face, and zoom out the character image into a normalized image, so that all the character images have the same length and width. Second, to select an image as a calibration image, calibration image the key feature point for the reference, the other images to a certain degree of rotation, zoom and translation, to locate all the key features of facial images, such as eyes, nose, corners of the mouth, eyebrows, and face parts contour points, etc. Finally, to correct and cut the character images after the alignment operation, to retain the effective part of the face carrying the key feature points of the face, and to keep the same length and width of all the images.

Figure 3

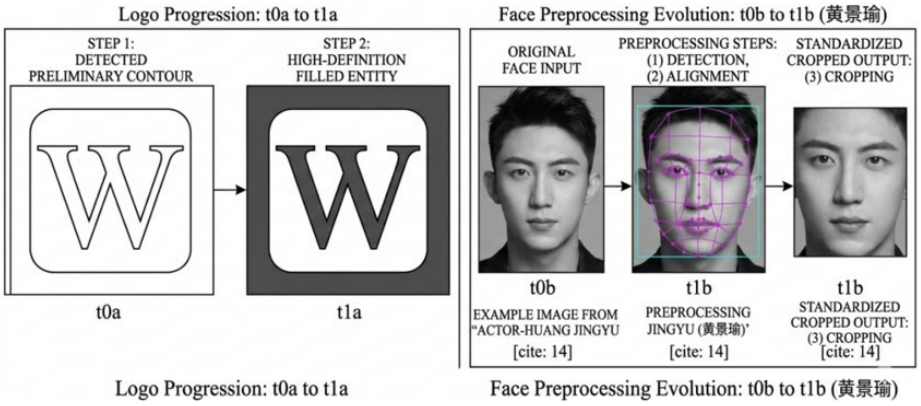

Figure 3 Image Preprocessing Process

2.2.3. Σ L SIMILARITY MATCHING

Similarity matching requires the existence of the benchmark image, and because the perceptual hash model is embedded in the malicious code as an attack component, the benchmark image should also be carried by the malicious code. In order to prevent the analyst from speculating the target image through the benchmark image, the attacker has the ability to customize the target image to ensure that the target image is not the target image and can meet the σ 1 similarity matching with the target image.

Figure 4 and Figure 5 show the rationality of this hypothesis, when setting α 1 to 0.7, corresponding to the reference plot of Figure 4(b)The image can be Figure 4(a), where the similarity in aHash, dHash and pHash calculations is higher than 0.7 and lower0.8; (a) (b) in Figure 5 only achieve a similarity match above 0.7 but lower than 0.8 in pHash calculation, but also provide the possibility for the attacker to make a special reference image. If the special reference image is a non-face image, the malicious code will have a higher accuracy when locating the target face, and it will increase the difficulty for the analyst to trace the target face image.

Figure 4

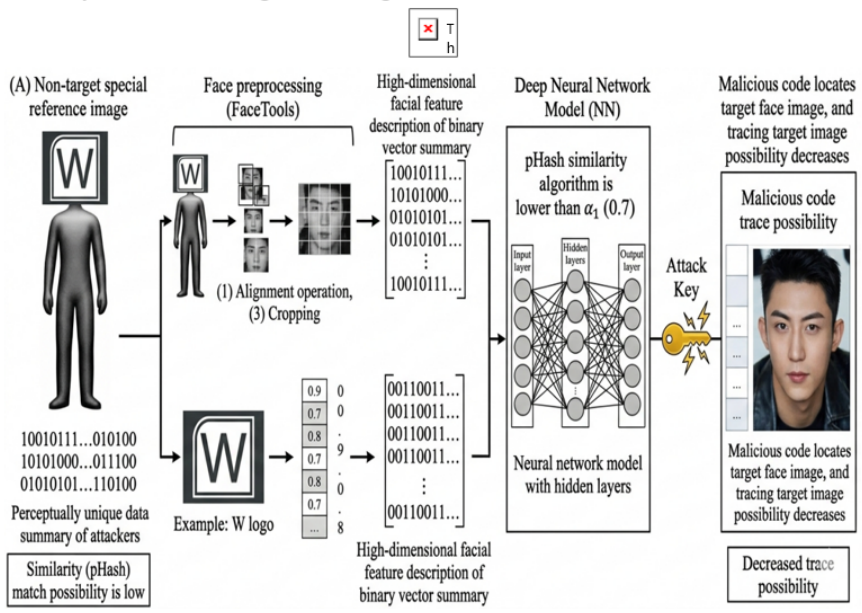

Figure 4 Schematic of 0.7 But Not 0.8

Figure 5

(A) Non-target image (b) target image

Figure 5 Phash Calculation Meets 0.7 But Not 0.8 Similarity Matching Diagram

The σ 1 similarity matching module receives the preprocessed image as the input image, requiring that the similarity between the input image and the special reference image is not less than σ 1. The σ 1 similarity matching algorithm presentation is shown in Table 1. To establish differentiation from other modules, the reference graph used in this module is called the global reference graph and represented in G; the input image is represented in P. By calculating the fingerprint of the image and the similarity measure of the fingerprint of P and G images, if the similarity between P and G is lower than σ 1, the image is the target image and is filtered out without any output; if the similarity between P and G exceeds σ 1, the valid local view of P is extracted and output, which is expressed in $Ploc_1$. This local view contains most of the key facial feature points and can be distinguished on the target character images and the non-target character images.

Table 1

Table 1 σ L Similarity Matching Algorithm
Input: Benchmark Global baseline plot G, to be perceived image P, σ 1. **Output:** the local view of the P $Ploc_1$; or, no output.
(1) ua=aHash(G), va=aHash(P);
(2) ud=dHash(G), vd=dHash(P);
(3) up=pHash(G), vp=pHash(P);
(4) Da=Hamminga(ua, va);
(5) Dd=Hamminga(ud, vd);
(6) Dp=Hamminga(up, vp);
(7) Psa=1-Da/n;
(8) Psd=1-Dd/n;
(9) Psp=1-Dp/n;
(10) if σ1≤Psa,Psd,Psp≤(σ1+ε), then
(11) $Ploc_1=P[x_0:x_1, y_0:y_1]$;
(12) return $Ploc_1$

Where, u and v are the binary representation of G and P image fingerprints, n is the binary length of the image fingerprint, D is the Hamming distance between u and v, ε is a defined threshold, [σ 1, σ 1 + ε] is the similarity interval between the constrained target image and G, and x and y are the horizontal and vertical coordinates of P, respectively. Moreover, to ensure that the first layer similarity matching has a low false positive rate, the module uses aHash, dHash, and pHash algorithms to bind σ 1 similarity matching.

2.2.4. Σ 2 SIMILARITY MATCHING

σ 2 similarity matching is based on the local matching of facial images. This module requires the existence of the local benchmark image, which is carried by malicious code.

As shown in Figure 6, a 160 160-size image can be split into 20 208 8 size images. Because local features are difficult to characterize global features, it is reasonable to assume that when the local reference image is small enough, the analyst cannot trace or restore the target face image. Therefore, in this work, the likelihood of tracing the global target image from local features is 0.

Figure 6

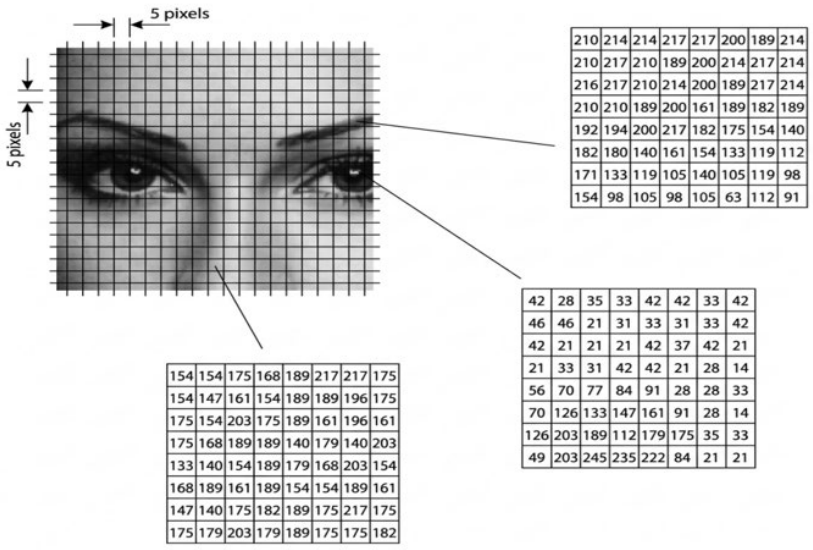

Figure 6 Schematic Diagram of the Image Segmentation

The σ 2 similarity matching algorithm presentation is shown in Table 4 and Table 3. The reference image of this module is called the local reference graph and is represented by L. σ 2 similarity matching module receives the $Ploc_1$ output by the σ 1 similarity matching module as the input image, requiring that the similarity between the input image and the corresponding special reference image should not be lower than σ 2. If the similarity between $Ploc_1$ and L is lower than σ 2, P is a non-target image, and the image is filtered without any output; if the similarity between $Ploc_1$ and L exceeds σ 2, P is the target image, extract and output two valid local views of P, which are expressed by $Ploc_2$ and $Ploc_3$, respectively. $Ploc_2$ and $Ploc_3$ are independent of $Ploc_1$ and often have more robust facial key features, enabling consistent hash fingerprints for all target person images.

Table 2

Table 2 The Σ 2 Similarity Matching Algorithm
Input: Benchmark Local baseline map L, to sense image $Ploc_1$, σ 2.
Output: two local views of P $Ploc_2$, $Ploc_3$; or, no output.
(1) up=pHash(L), vp=pHash($Ploc_1$);
(2) Dp=Hamminga(up,vp);
(3) Psp=1-Dp/n;
(4) if Psp≥σ 2,then
(5) $Ploc_2$=P[x_2: x_3, y_2:y_3];
(6) $Ploc_3$=P[x_4:x_5, y_4:y_5];
(7) return $Ploc_2$,$Ploc_3$

Where u and v are the binary representations of two images corresponding to L and $Ploc_1$, n is their binary length, D is the Hamming distance between u and v, Ps is the similarity of u and v, and x and y are the horizontal and ordinate of P, respectively.

2.2.5. KEY GENERATION

The key generation module is a third layer similarity match with a similarity threshold of 1. As shown in Table 4, this module receives the $Ploc_2$ and $Ploc_3$ output of the σ 2 similarity matching module as input and calculates the image fingerprints of both based on the pHash algorithm, represented by pHash2 and pHash3, respectively. The two image fingerprints are consistent for all the target character images, and the length of each fingerprint string is 64 bits. The splicing result of the two is output as the addition / decryption key of the malicious attack load, and the binary length is 128 bits. pHash2 and pHash3 can generate the right key for the correctness, so the difficulty of violence is 264 264, or 2128.

Table 3

Table 3 The Key Generation Algorithm
Input: two local views of P $Ploc_2$, $Ploc_3$.
Output: key.
(1) pHash2=pHash ($Ploc_2$), pHash3=pHash ($Ploc_3$);
(2) key=pHash2 + pHash3;
(3) return key.

3. RESULTS AND DISCUSSION

3.1. EXPERIMENTAL SETTING

This work crawled the target people and the remaining 25 non-target face images based on the Google map bed and downloaded 77 non-target faces from the VGGFace dataset. In addition, from the perspective of the simulated attack, this work by default assumes that the image captured through the camera should be a positive face image. This assumption is reasonable; usually, the face image captured by the camera is required when the user logs on to the computer through facial recognition. Carrying a key and correct facial feature requires facial recognition, while non-positive face images often lose some of the key features, resulting in facial recognition failure. Therefore, this work screened non-positive face images from the crawling and downloaded dataset and finally retained 75 positive face images of specific people and 786 frontal face images of multiple non-target people, which contained 66 frontal face images of different people in the VGGFace dataset. In order to evaluate the performance of our model, we utilized two different metrics as illustrated in Eqs. 1 and 2.

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

To test the performance of the perceptual hash model on all positive and negative samples, and the test results are shown in Table 4

Table 4

Table 4 Statistics of Similarity Matching Pass Rate Test Results			
Core function		**Target character image**	**Non-target person images**
target recognition	σ 1 Similarity	100%	0.13%
	Match		
	σ 2 Similarity	97.30%	0%
	Match		
	Key generation	96%	0%

Corresponding to the confusion matrix, the values of TP, FP, TN, and FN are 72, 0, 786, and 3, respectively, so that the perceptual hash model can achieve 100% accuracy and the recall rate is 96%. This result successfully validates the perceptual hash model

Of effectiveness, which is consistent with Deep Locker in the implementation of core function and even slightly better than Deep Locker in recall.

3.2. VOLUME INCREMENT COMPARISON EXPERIMENT

Volume increment refers to the volume change before and after its embedding in the deep neural network model or the perceptual hash model during the process of constructing the secret and accurate malicious code. Considering the impact of deep learning-related open-source libraries on the volume of deep learning models. Comparing the deep neural network model constructed in two different ways with the perceptual hash model proposed in this work. One is a 0 to 1 implementation of B-DNN, the model volume is only 1850 KB; the other is a deep neural network model based on the TensorFlow open-source library tf-DNN.

It represents a class of AI models built based on universal open-source libraries.

This work divides the volume increment resulting from the embedded model into three parts: (1) code volume increment; (2) model volume increment; (3) volume increment of the benchmark image.

First, assuming the volume of the original malicious code is a, in the original malicious code fragments to join the implementation of the model, function library call, model call, and other related code generated after the new malicious code volume is b, because the code volume increment as b-a, usually a function library call will lead to a larger code volume increment. Second, the neural network structure and neural network parameters are called by malicious code, which are binary files that need to be embedded in malicious code. Correspondingly, the perceptual hash model is a code-for-a implementation way, so there is no model volume increment. Finally, the malicious code implemented based on the perceptual hash model needs to be embedded with two benchmark images, and the sum of the volume of the two images is the volume increment of the benchmark image. Correspondingly, the malicious code implemented based on the deep neural network model does not need to carry the benchmark image, so there is no such volume increment present.

Table 5

Table 5 Comparison of Model Volume Increments

Model	tf-DNN	B-DNN	Perceptual hash model
Code volume increment	360.5MB	2524KB	2522KB
Model volume increment	293.2MB	1085KB	0MB
Reference image volume increment	0MB	0MB	8.9KB
Total volume increment	653.7MB	4.27MB	2.47MB

In conclusion, the volume increment of the three models is shown in Table 5. tf-DNN is obviously not practically feasible because of the obvious volume increment. The volume increment brought by the B-DNN and the perceptual hash model is relatively small, and the perceptual hash model has a higher light load advantage when actually building malicious code.

3.3. PRACTICAL FEASIBILITY COMPARISON EXPERIMENT

The tf-DNN, B-DNN, and perceptual hash model are used to compare different deep neural network models and perceptual hash models in terms of startup time consuming and memory footprint. In each experiment, the three models were successively separately on 8 virtual machines, and the starting time and memory usage data of the model were recorded. Thirty experiments were repeated to record the performance of the three models on different nodes.

The startup-time results show that the perceptual hash model has the shortest and most stable startup time. As the number of experiments increases, the startup time of the B-DNN model tends to stabilize and remains close to that of the perceptual hash model, indicating little noticeable delay in practical use. However, the B-DNN model occasionally exhibits sharp spikes in startup time on the master node and several slave nodes (rising from nearly 0 s to 3.5 s), mainly

because a large number of model parameters must be reloaded into memory. By contrast, the perceptual hash model produces smoother and more stable startup-time data, suggesting higher practical feasibility.

Secondly, compared with the perceptual hash model and the B-DNN model, the starting time of the tf-DNN model is generally higher than 100s, and the occasional time increase will also reach more than 200s, which will have an obvious time-consuming feeling for human experience. This high time consuming is mainly due to the loading time of tensorflow open-source library. Therefore, a class of AI model based on open-source library represented by tf-DNN will have low practical feasibility if it is used to help implement attacks in the unknown defense space.

The memory-usage results show that, except for the slave5 node, the perceptual hash model generally maintains a memory occupancy rate of about 1.7%–1.8%.In summary, according to the startup time and memory usage of the above three models, tf-DNN has high startup time and memory usage, low practical feasibility, and cannot well help phMalware to complete the attack process, while the experimental data of perceptual hash model and B-DNN model are very close, both have high practical feasibility. Further, according to the above experimental data, perceptual hash model in terms of time consuming and memory occupancy with numerical performance and data stability advantages, to verify the correctness of the phMalware on the technology selection, not only realized the major breakthrough of DeepLocker, also further emphasizes the new secret precision malicious code security threats.

4. DISCUSSION

Experiments show that phMalware has functional advantages, for instance, accurate identification of attack targets, concealment of attack intentions, and generalization consistency while maintaining performance advantages such as light load training-free and small size. phMalware is an effective solution for AI-poweredenvironment-keying malware that is difficult to be exploited in the wild, which is a key step for this kind of malware to move from the laboratory to practice. To prevent such malware from being maliciously used by an attacker, we focus on the defense methods against such attacks.

4.1. GRAPH ANALYSIS METHOD BASED ON THE BEHAVIORAL SEQUENCE

Based on the aforementioned threat analysis of phMalware, its attack characteristics can be summarized as follows: (1) In order to ensure the secrecy of the target image in the input space, the delivery of phMalware should be arbitrarily distributed, and the process of accurate target positioning will involve a large number of (non-target) image reading, image processing, and other operations. (2) Progressive hash calculation is the decisive process of obtaining the key of the unique environment, while the perceptual hash is usually used for similarity detection and rarely sees progressive use. Therefore, this is an obvious characteristic point of phMalware. (3) Malicious code has built-in two benchmark images. All hash calculations need to be fingerprint matched with at least one of the benchmark images, and the matching is bound to the image extraction operations by a certain extent.

Separate detection for any of the above features is not enough to use as outlier to identify malicious code. However, the above characteristics can be captured by behavioral monitoring or reverse analysis. As shown in Figure 7 the method based on graph analysis can build a distinct behavior sequence, which can be used as the feature point for malicious code detection.

Figure 7

Figure 7 Phmalware

4.2. HETEROGENEOUS DEFENSE METHOD BASED ON BEHAVIOR MONITORING

phMalware as a kind of environment-typed malicious code, depends on the early information collection of the target environment, for example, through the collection of face image information, to help build the perception hash model; through the collection of target defense environment information, to help build accurate and effective attack load. A heterogeneous defense model that uses behavior monitoring can be set up on the security side of the target environment. Figure 8 shows that a behavior monitor collects information in the defense environment. When key environmental

information is collected, the behavior monitor will perceive it and start a heterogeneous update of the target defense environment. This makes the information the attacker collected useless. Also, if an attacker tries to deliver malicious code, the new defense increases the chances that the attack will fail.

Figure 8

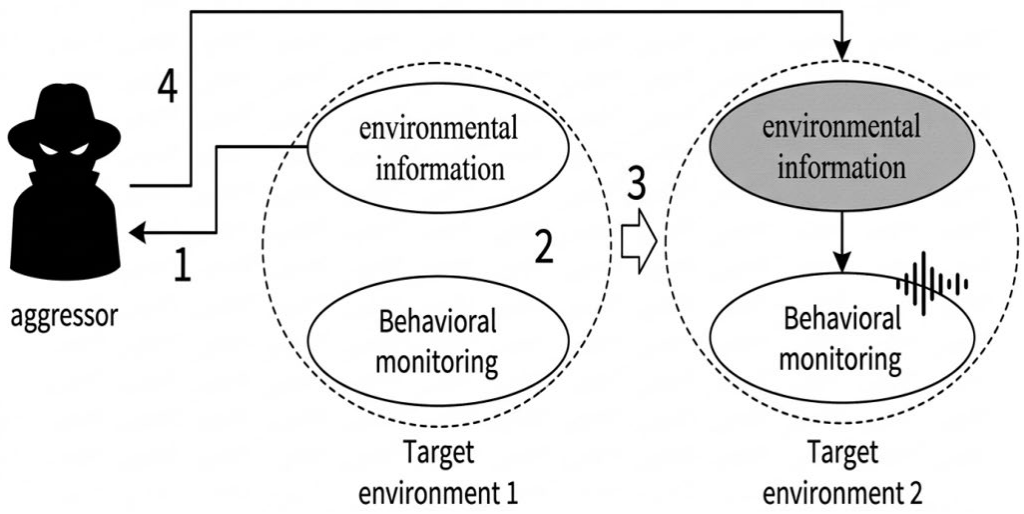

Figure 8 Heterogeneous Defense Based on Behavioral Monitoring

4.3. DEFENSE METHOD BASED ON PERCEIVED HASH COLLISION

The core component of phMalware is the perceptual hash model. Under the premise of satisfying the secrecy of the target feature and the secrecy of the decryption key, the main way to defend phMalware is to crack the perceptual hash model, that is, to create two hash same but completely different images through hash collision. For example, NeuralHash is a perceptual hash function using neural network, and researchers use the gradient descent properties of neural network to discover hash conflicts, proving that

NeuralHash is susceptible to the influence of adversarial samples Struppek et al. (2022).

Although aHash, dHash, and pHash have not been found through a large number of literature research, the perceptual hash collision study for NeuralHash algorithm provides the possibility to crack the perceptual hash model. Therefore, it is believed that the defense method based on the perceived hash collision will be an important research scheme in the future. If the decryption key can be successfully obtained through the perceived hash collision, the malicious code will carry malicious intent Biswas et al. (2017).

phMalware needs to silently execute image processing, fingerprint calculation and other related operations in the background. Defenders can focus on the detection of suspicious processes running in the background and achieve the purpose of security defense by checking and killing suspicious processes. Monitor any attempts to capture faces through the camera and track the main program responsible for this action. If an untrusted program initiates the operation, trigger a security warning right away and either remove or isolate the untrusted program Yu (2020). phMalware requires two benchmark images, but designing these images often takes many experiments. Defenders can try to predict that attackers will use low-entropy benchmark images. By using experience or limited brute-force methods, security teams can narrow down the possible target images. If the malicious intent cannot be uncovered through reverse engineering or other methods, the signatures of certain malware components can help detect malicious code. Even though phMalware uses different environment keys to encrypt its attack payloads for different targets, it still relies on common components, such as image processing, progressive hash calculation, image matching, and image extraction. By identifying the signatures of these shared components, security teams can collaborate across networks to detect and defend against this type of malware.

5. CONCLUSION

As AI and modern cryptography become more integrated, network attack and defense are changing quickly. To address the high resource use and deployment challenges posed by new intelligent malware, this paper explains the intelligent malware attack chain and clearly defines the IUM threat model. It also introduces perceptual hash technology for environmental key generation, leading to the creation of the phMalware model. Tests show that the generalized consistency perceptual hash model can fully replace large DNN models. It delivers precise target attacks, hides its intentions well, and uses much less payload and resources. This means that lightweight, stealthy intelligent malware is now ready for real-world deployment, not just lab tests. Traditional security systems struggle to keep up with this new threat. The paper discusses new defense ideas, such as analyzing behavior sequences with graphs, updating defenses in different environments, and breaking perceptual hash collisions. Future work will examine how to add hash collision detection and behavior sequence graph analysis to real-time operating system kernels in a cost-effective and efficient manner, with the goal of stopping IUM intelligent malware attacks at their source.

CONFLICT OF INTERESTS

None.

ACKNOWLEDGMENTS

None.

REFERENCES

Aissi, M. B., El Hazzat, S., and Yakhlef, M. B. (2024). Evaluation of Perceptual Hashing Algorithms for Digital Image Authenticity. In 2024 3rd International Conference on Embedded Systems and Artificial Intelligence (ESAI) (1–7). IEEE. https://doi.org/10.1109/ESAI62891.2024.10913543

Asghar, H. J., Zhao, B. Z. H., Ikram, M., Nguyen, G. L. D., Kaafar, D., Lamont, S., and Coscia, D. (2024). Use of Cryptography in Malware Obfuscation. Journal of Computer Virology and Hacking Techniques, 20(1), 135–152. https://doi.org/10.1007/s11416-023-00504-y

Barchuk, B., and Volkov, K. (2025). Antivirus Evasion Techniques in Modern Malware. World Journal of Advanced Research and Reviews, 26(2). https://doi.org/10.30574/wjarr.2025.26.2.1966

Bhaskara, V. S., and Bhattacharyya, D. (2018). Emulating Malware Authors for Proactive Protection Using GANs Over a Distributed Image Visualization of the Dynamic File Behavior. arXiv.

Biswas, R., Fidalgo, E., and Alegre, E. (2017). Recognition of Service Domains on TOR Dark Net Using Perceptual Hashing and Image Classification Techniques. In 2017 8th International Conference on Imaging for Crime Detection and Prevention (ICDP) (7–12). Institution of Engineering and Technology. https://doi.org/10.1049/ic.2017.0034

Chawande, S. (2024). AI-Driven Malware: The Next Cybersecurity Crisis. World Journal of Advanced Engineering Technology and Sciences, 12(1), 542–554. https://doi.org/10.30574/wjaets.2024.12.1.0172

Chen, Z., and Cao, J. (2023). VMCTE: Visualization-Based Malware Classification Using Transfer and Ensemble Learning. Computers, Materials and Continua, 75(2), 4445–4465. https://doi.org/10.32604/cmc.2023.038639

Ding, Y., Zhang, M., Li, X., Chen, X., Zhu, J., Hu, W., and Shan, C. (2024). Defending Against Malicious Code: A Comprehensive Study of Evasion and Mitigation Techniques. In Computational and Experimental Simulations in Engineering (376–389). Springer. https://doi.org/10.1007/978-3-031-77489-8_29

Dos Santos, H., Martins, T. S., Barreto, J. A. D., Nakamura, L. H. V., Ranieri, C. M., De Grande, R. E., Filho, G. P. R., and Meneguette, R. I. (2024). ChaSAM: An Architecture Based on Perceptual Hashing for Image Detection in Computer Forensics. IEEE Access, 12, 104611–104628. https://doi.org/10.1109/ACCESS.2024.3435027

Guo, Q., Rungsrisawat, S., and Liu, C. (2025). The Future of Talk Shows: AI-Driven Virtual Hosts and Their Impact on Media Communication: A Systematic Literature Review. Journal of Advances in Humanities Research, 4(2), 37–57. https://doi.org/10.56868/jadhur.v4i2.306

Jain, S., Creţu, A.-M., Cully, A., and de Montjoye, Y.-A. (2023). Deep Perceptual Hashing Algorithms with Hidden Dual Purpose: When Client-Side Scanning Does Facial Recognition. In 2023 IEEE Symposium on Security and Privacy (SP) (234–252). IEEE. https://doi.org/10.1109/SP46215.2023.00014

Koch, L., and Begoli, E. (2025). Adversarial Binaries: AI-Guided Instrumentation Methods for Malware Detection Evasion. ACM Computing Surveys, 57(5), Article 108. https://doi.org/10.1145/3706573

Nadeem, A., Rimmer, V., Joosen, W., and Verwer, S. (2022). Intelligent Malware Defenses. In L. Batina, T. Bäck, I. Buhan, and S. Picek (Eds.), Security and Artificial Intelligence: A Crossdisciplinary Approach (217–253). Springer. https://doi.org/10.1007/978-3-030-98795-4_10

Nguyen, T., McDonald, J. T., Glisson, W. B., and Andel, T. R. (2020). Detecting Repackaged Android Applications Using Perceptual Hashing. In Proceedings of the 53rd Hawaii International Conference on System Sciences (1–10). https://doi.org/10.24251/HICSS.2020.813

Riordan, J., and Schneier, B. (1998). Environmental Key Generation Towards Clueless Agents. In G. Vigna (Ed.), Mobile Agents and Security (15–24). Springer. https://doi.org/10.1007/3-540-68671-1_2

Shahid, A. (2025). Exploring the Hashtag Movements Role on Facebook in Political Activism Among Youth of Pakistan: A Digital Ethnography. International Journal of Management Thinking, 3(1), 1–21. https://doi.org/10.56868/ijmt.v3i1.81

Struppek, L., Hintersdorf, D., Neider, D., and Kersting, K. (2022). Learning to Break Deep Perceptual Hashing: The Use Case Neuralhash. In Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (58–69). Association for Computing Machinery. https://doi.org/10.1145/3531146.3533073

Yu, J. (2020). Massive Malware Variants Detection Based on Bag-of-Words Perceptual Hashing. Journal of Physics: Conference Series, 1682(1), Article 012046. https://doi.org/10.1088/1742-6596/1682/1/012046