中國科學院軟件研究所
Institute of Software, Chinese Academy of Sciences

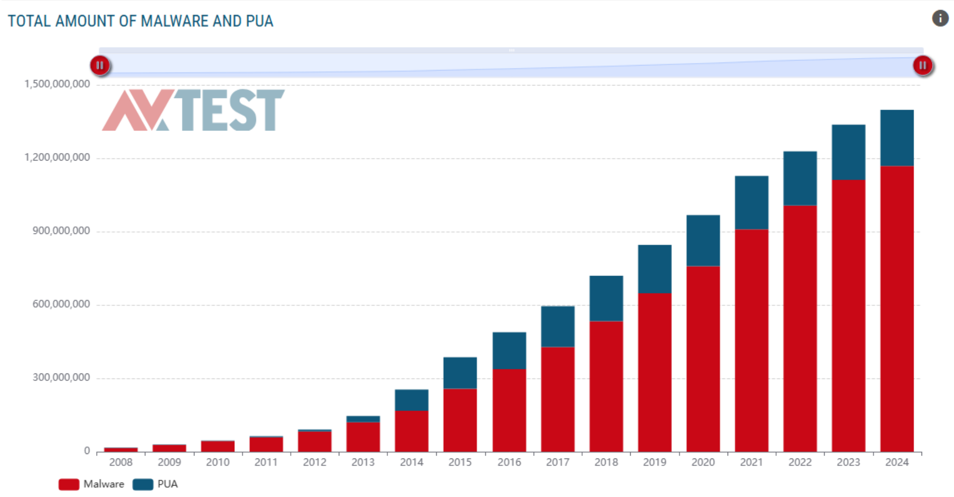
A Wolf in Sheep's Clothing: Practical Black-box Adversarial Attacks for Evading Learning-based Windows Malware Detection in the Wild

33rd USENIX Security Symposium 2024

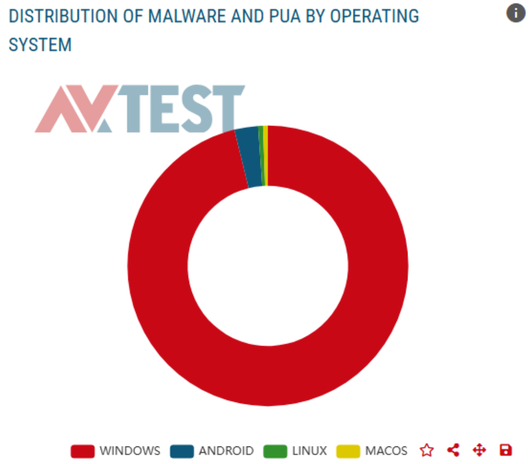
Xiang Ling, Zhiyu Wu, Bin Wang, Wei Deng, Jingzheng Wu, Shouling Ji, Tianyue Luo, Yanjun Wu

Windows Malware

- **Malware remains one of the most serious security threats**
  - Normally perform malicious activities on computer systems
    - stealing sensitive information
    - demanding a large ransom
    - disrupting national infrastructures
    - ……



- **Windows Malware**
  - World-wide popularity of Windows operating systems
  - Windows has becoming the main target of malware attackers



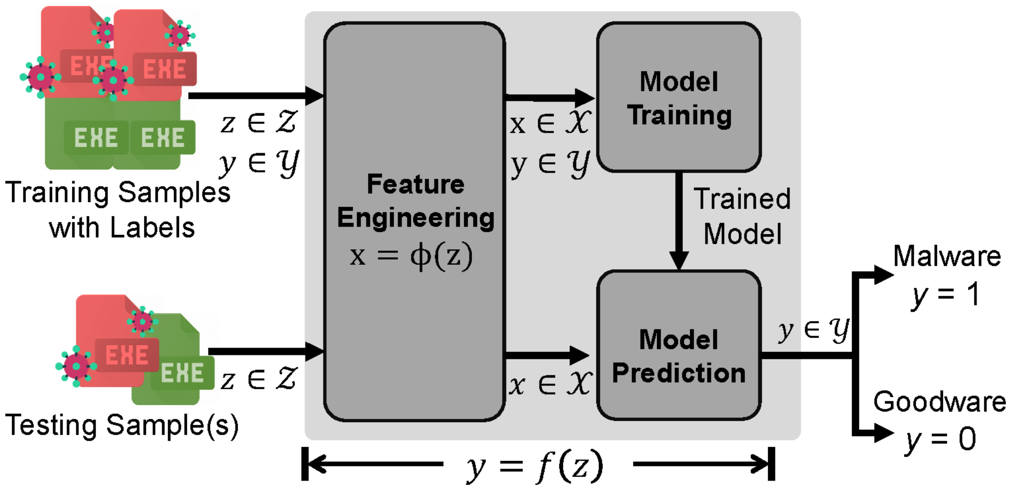
Source: AV-TEST Institute https://portal.av-atlas.org (July 12, 2024)

Windows Malware Detection and Anti-virus Products

- **Malware detection:** static analysis *versus* dynamic analysis
- For Windows malware, the static-analysis-based detection can be generally categorized into:
  - **Signature-based malware detection**
    - fast speed in detecting malware
    - cannot detect previously unknown malware
    - easily evaded by obfuscations like compression, register reassignment, code virtualization, etc.
  - **Learning-based malware detection**
    - leverage the high learning capability of machine learning / deep learning models
    - can detect some newly emerging malware
    - make some obfuscations ineffective for evasions
- More and more mainstream anti-virus products (Kaspersky, McAfee, etc.) adopt the learning-based malware detection as a pivotal component in their security solutions

How about the security risk of learning-based Windows malware detection?

- **Target model:** the learning-based Windows malware detection $\boldsymbol{f}(\cdot)$
- **Adversary's goal**
  - Misclassify malware as goodware
  - Preserve the original semantics
- **Adversary's knowledge & capability**
  - Classic black-box adversarial attack
    - **scenario #1**: black-box attack with predicted probabilities
    - **scenario #2**: black-box attack without predicated probabilities
  - No prior information on the target model
    - no training dataset
    - no extracted feature set
    - no learning algorithm with parameters
    - no model architectures with weights
    - ……
  - Adversary has the capability of manipulating Windows executables while adhering to its standard specifications



MalGuise: Overall Framework

- **Two challenges:**
  - **How to generate the adversarial malware file?**
    - maintain the same semantics as the original one
    - remain less noticeable to possible defenders
  - **How to efficiently search the adversarial malware?**
    - search in the large & discrete space of malware
    - search in the strict black-box setting

- **Our solution: MalGuise**
  ① Adversarial Transformation Preparation
  ② MCTS-Guided Searching
  ③ Adversarial Malware Reconstruction

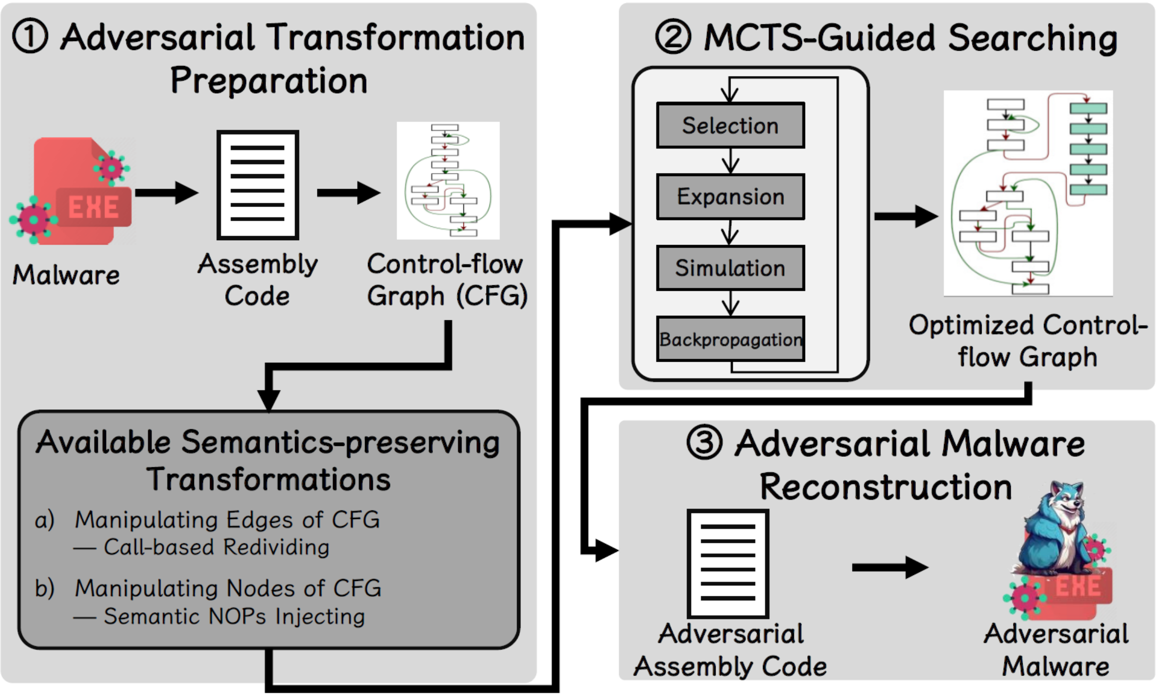
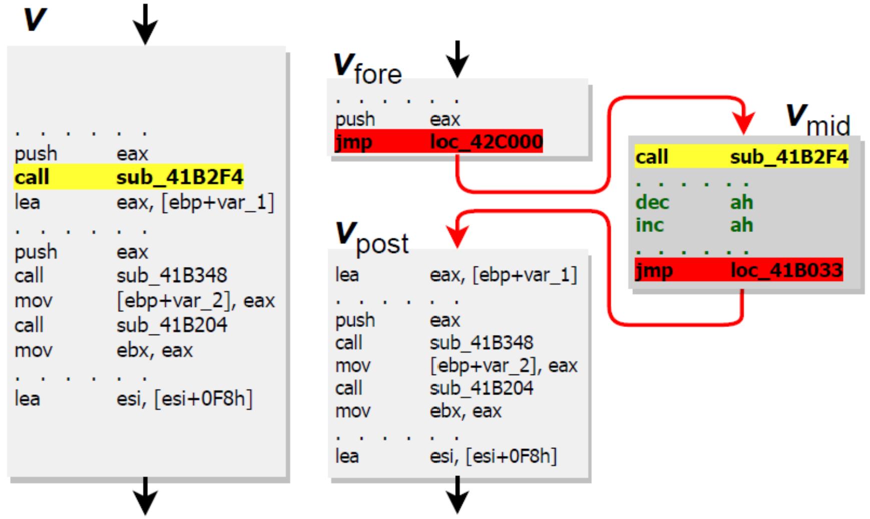


Figure 2: The overview framework of MalGuise.

MalGuise: ① Adversarial Transformation Preparation

- Represent the given malware as control-flow graph (CFG)
- Present a novel semantic-preserving transformation of **call-based redividing**
  - ➢ annotate all available basic blocks having the call instruction
  - ➢ select one call instruction in the basic block as the dividing line
  - ➢ redivide the basic block V as a combination of three basic blocks
    - □ $V \rightarrow \{V_{fore}, V_{mid}, V_{post}\}$
  - ➢ enrich V_{mid} by injecting semantic NOPs



(a) Before transformation. (b) After applying a `call-based redividing`.

Figure 3: The `call-based redividing` redivides one basic block in the "LockBit 3.0" ransomware (*i.e.*, Fig. 3(a)) into a composite of three consecutive basic blocks (*i.e.*, Fig. 3(b)).

MalGuise: ② MCTS-Guided Searching

- Optimizing a sequence of call-based redividing transformations, i.e., $\mathbf{T} = T_1 \odot T_2 \dots \odot T_n$
  - T_i is one atomic call-based redividing and involves two decision-markings:
    - select one from all available call instructions to be redivided
      - every call can be repeated selected in a recursive manner
    - determine the proper semantic NOPs to be injected
      - semantic NOPs can be infinitely generated with context-free grammar
- MCTS-guided searching algorithm
  - input: the given malware's CFG, i.e., x
  - output: the transformation sequence $\mathbf{T}$
  - Monte-Carlo tree searching based optimization
    - widely used to solve long-standing optimization problems
    - requires little or no domain knowledge

Algorithm 1: MCTS-Guided Searching Algorithm.

Input : a given malware z with its CFG x, target system f, max length N, simulation number S, budget C.
Output : the transformation sequence $\mathbf{T}$.

```
Begin
    I^call ← GetAllCalls(x);
    v, T ← InitMCTSRootNode(x, I^call), ∅ ;    //initialize
    for i ← 1 to N do              //loop upto maximum length
        for j ← 1 to C do    //loop upto computation budget
            if random(0,1) < 0.5 then //avoid unlimited expansion
                v_selected ← Selection(v);
            else
                v_selected ← Expansion(v);
            reward ← Simulation(v_selected, f, S);
            BackPropagation(v_selected, reward);
        v_node ← ChildWithHighestReward(v);
        T ← T.append(v_node.T);
        x_adv ← v_node.x;
        if Evaded(f, x_adv) == True then
            return T
        v ← v_node
```

MalGuise: ③ Adversarial Malware Reconstruction

- Reconstruct the final adversarial malware file z_{adv}
- Requirements:
  - adhere to the specifications of Windows executables
  - avoid unexpected errors, e.g., addressing errors
- Solutions:
  - for each call-based redividing transformation, patch the malware file by injecting the V_{mid} into the slack space or the new section
  - adjust other fields in the header
  - ……

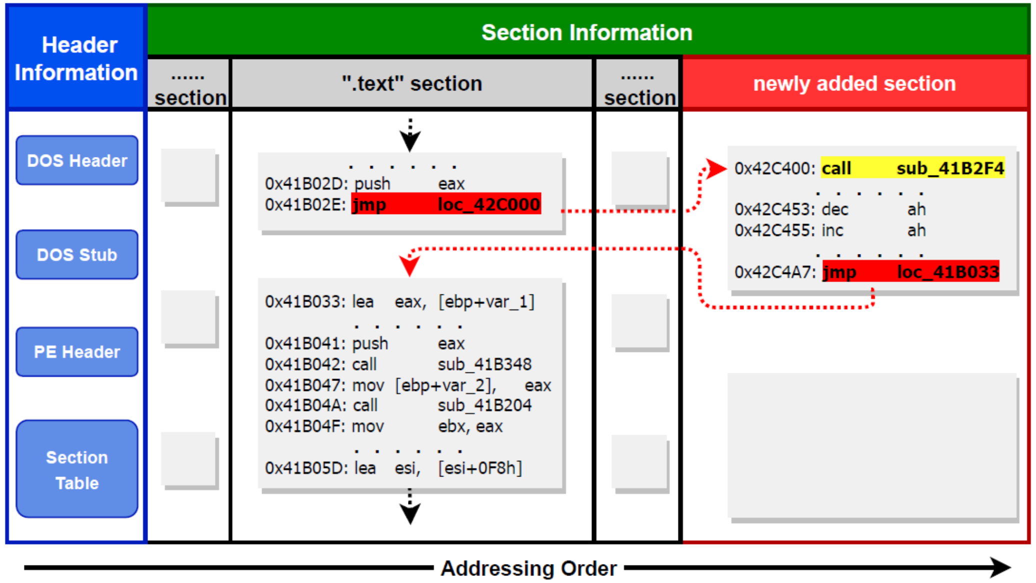


Figure 4: The conceptual layout of the reconstructed adversarial Windows malware file for the "LockBit 3.0" ransomware.

Evaluation Settings

- **Benchmark dataset**
  - a balanced dataset of 210,251 Windows executables
  - split into three disjoint training/validation/testing datasets
- **Target systems with detecting performance**
  - learning-based Windows malware detection systems
    - MalGraph (INFOCOM 2021)
    - Magic (DSN 2019)
    - MalConv (arXiv 2017, citations~670)
  - anti-virus products
    - McAfee, Comodo, Kaspersky, ClamAV, Microsoft Defender ATP
- **Baseline attacks**
  - Two adversarial attacks:
    - MMO (Lucas et al., Asia CCS 2021)
    - SRL (Zhang et al. TDSC 2022)
  - Three obfuscation tools: UPX, VMProtect, Enigma

Table 1: Summary statistics of the benchmark dataset.

Dataset	Training	Validation	Testing	Total
Malware	81,641	10,000	10,000	101,641
Goodware	88,610	10,000	10,000	108,610
Total	170,251	20,000	20,000	210,251

Table 2: The detecting performance of three learning-based Windows malware detection systems in our testing dataset.

Target Systems	AUC (%)	FPR = 1%		FPR = 0.1%	
		TPR (%)	bACC (%)	TPR (%)	bACC (%)
MalGraph	99.94	99.34	99.18	92.78	96.36
Magic	99.89	99.02	99.02	89.28	94.59
MalConv	99.91	99.22	99.12	86.54	93.22

Answer to RQ1 (Attack Performance)

RQ1 (Attack Performance): What is the attack performance of MalGuise against the state-of-the-art learning-based Windows malware detection systems?

- **Evaluation setup:**
  - two black-box scenarios
  - two kinds of baseline attacks
- **For baseline adversarial attacks:**
  - MMO shows inferior attack performance on all target models in both scenarios
  - SRL shows obviously higher ASRs against Magic
- **For baseline obfuscation tools:**
  - all three obfuscations show inferior attack performance
  - VMProtect achieves the worst attack performance as it typically obfuscate a small portion of the malware file

Table 3: The ASR performance (%) comparisons between MalGuise and baseline attacks against three target systems under two black-box scenarios, *i.e.*, *w/ prob.* and *w/o prob.*

Black-box Scenarios	Attacks	MalGraph FPR =1%	MalGraph FPR =0.1%	Magic FPR =1%	Magic FPR =0.1%	MalConv FPR =1%	MalConv FPR =0.1%
w/ prob.	MMO	15.55	52.30	12.82	40.13	11.99	39.66
	SRL	2.39	19.59	25.38	86.77	—	—
	MalGuise	**97.47**	**97.77**	**99.29**	**99.42**	**34.36** (**97.76**)	**97.38** (**99.77**)
w/o prob.	MMO	3.73	27.83	3.41	25.46	2.46	20.72
	SRL	2.59	15.28	3.84	47.48	—	—
	UPX	0.55	4.43	3.30	39.80	0.31	9.32
	VMProtect	0	0	0.23	4.33	0	0
	Enigma	0.81	11.69	0	28.96	0	0.24
	MalGuise	**96.84**	**96.49**	**99.27**	**99.07**	**31.41** (**95.18**)	**88.02** (**99.77**)

"—" means SRL does not apply to MalConv as it cannot generate real malware files.

- MalGuise achieves the best attack performance on all target models in both scenarios

Answer to RQ2 (Utility Performance)

RQ2 (Utility Performance): Does the adversarial malware generated by MalGuise maintain the original semantics?

Evaluation setup:

- SPR = the ratio of adversarial malware files that preserve the original semantics among all generated adversarial malware files
- no exact solution to judge $Sem(z, z_{adv})$ due to the inherent complexity of executable
- present an empirical solution by collecting and comparing the two API sequences invoked when they are run on the same sandbox

$$SPR = \frac{|Sem(z, z_{adv}) = 1|}{|(f(z) = 1) \wedge (f(z_{adv}) = 0)|}, \forall z \in \mathbf{Z}$$

$$Sem(z, z_{adv}) = \begin{cases} 1 & \text{if } dist_{norm}(z, z_{adv}) < dist_{\Delta} \\ 0 & \text{otherwise.} \end{cases}$$

$$dist_{norm}(z, z_{adv}) = \frac{Distance(\mathtt{API}_z, \mathtt{API}_{z_{adv}})}{max(l(\mathtt{API}_z), l(\mathtt{API}_{z_{adv}}))} \in [0, 1]$$

Evaluation results:

- SRL is not applicable as it generates adversarial features
- for MMO, only less than 50% of adversarial malware preserves their original semantics
- MalGuise achieves the best utility performance with over 91% of generated adversarial malware preserving their original semantics

Table 5: The SPR (%) of MalGuise and two baseline adversarial attacks against three target systems.

Attacks	MalGraph		Maigc		MalConv	
	FPR=1%	FPR=0.1%	FPR=1%	FPR=0.1%	FPR=1%	FPR=0.1%
MMO	41.8	49.4	39.6	39.8	39.2	50.8
SRL	—	—	—	—	—	—
MalGuise	**91.84**	**91.99**	**93.45**	**92.28**	**92.67**	**91.68**

Answer to RQ3 (Real-world Performance)

RQ3 (Real-world Performance): To what extend does MalGuise evade existing commercial anti-virus products?

- For 4/5 evaluated anti-virus products, MalGuise achieves over 30% ASRs, presenting potential tangible security concerns to real-world users
- MalGuise can be further improved by carefully fine-tuning its hyper-parameters, e.g., limit the semantic NOPs to 25 most effective opcodes, MalGuise(S)
- MalGuise can be applied against anti-virus products by only modifying very few blocks in CFG
  - for McAfee, Comodo and ClamAV, over 90% adversarial malware only need to modify one basic block
  - the other two anti-virus products (i.e., Kaspersky & MS-ATP) only need to modify two basic blocks

Table 6: The ASR (%) of MalGuise against five anti-viruses.

Attacks	McAfee	Comodo	Kaspersky	ClamAV	MS-ATP
MalGuise	48.81	36.00	11.29	31.94	**70.63**
MalGuise(S)	52.49	36.36	13.36	32.33	**74.97**
Increased ASR	+3.68	+0.36	+2.07	+0.39	+4.34

Table 7: Distribution frequency (%) of the number of modified blocks for adversarial malware that evades anti-virus products.

# of blocks	McAfee	Comodo	Kaspersky	ClamAV	MS-ATP
1	96.66	94.28	88.17	97.54	38.21
2	4.58	4.71	9.68	2.05	42.88
3	0.76	1.01	2.15	0.41	17.35
4	0	0	0	0	1.17
5	0	0	0	0	0.39

Conclusion

- To understand and evaluate the security risks of existing learning-based Windows malware detection, we propose a practical black-box adversarial attack framework of MalGuise
- MalGuise is the first to apply a fine-grained manipulation towards the CFG representation of Windows executables, which not only manipulates the nodes of CFG but also its edges
- Evaluations show that MalGuise not only effectively evades state-of-the-art learning-based Windows malware detection with attack success rates exceeding 95%, but also evades five anti-virus products, achieving attack success rates ranging from 11.29% to 74.97%
- Code sharing to verified academic researchers at https://github.com/jiyuay/MalGuise-Access-Instructions

Thanks for Listening!

For any questions, feel free to contact
e-mail: lingxiang@iscas.ac.cn
homepage: ryderling.github.io