BAR 2023

FCGAT: Interpretable Malware Classification Method using Function Call Graph and Attention Mechanism

March 3, 2023

Minami Someya[*†] Yuhei Otsubo[†*] Akira Otsuka[*]

* Institute of Information Security, Japan

† National Police Agency, Japan

Malware Classification

- Identifies malware family or category

Emotet, Trickbot …

Downloader, Ronsomware …

- Usefulness of malware classification:
  - Understands malware behavior
  - Helps with malware analysis

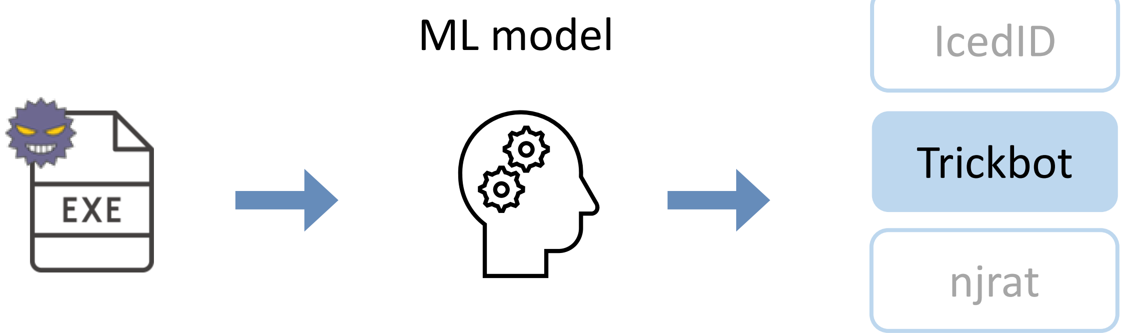
- Limitations of conventional signature-based methods:
  - Cannot keep up with creating pattern files of new malware

- Solution:
  - Use machine learning to classify malware

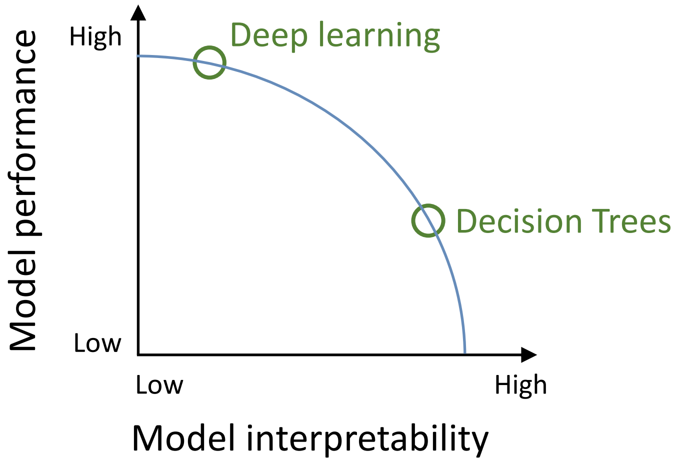
Drawbacks in ML Method: Lack of Interpretability

- ML-models are often treated as black boxes

Why Trickbot?
Where’s the malicious code?



- Model interpretability and performance are often in a trade-off relationship



Overview

IISEC
Insitute of Information Security

Research Goals

- Creating classifier that can explain the reasons for malware classification
- Achieving both high classification performance and interpretability

Solution

- FCGAT: Interpretable Malware Classification Method
  using **F**unction **C**all **G**raph and **AT**tention Mechanism

Contributions

- Successfully classified malware families with high performance comparable to cutting-edge methods
- Analyzed the explanations and obtained insight into the functions that characterize malware

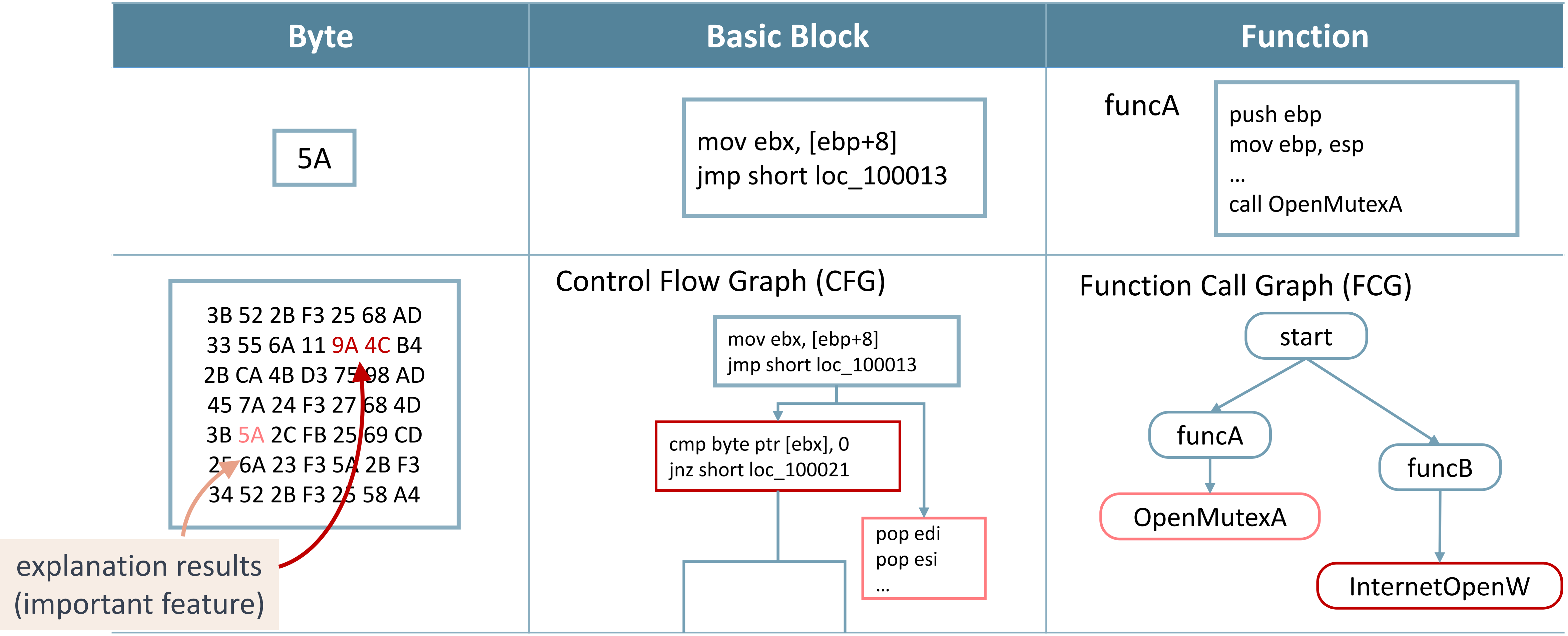
Determining Feature Set

Byte	Basic Block	Function
5A	mov ebx, [ebp+8] jmp short loc_100013	funcA push ebp mov ebp, esp … call OpenMutexA

Which feature set should we use?

We need to consider the interpretability of explanation results.

Determining Feature Set



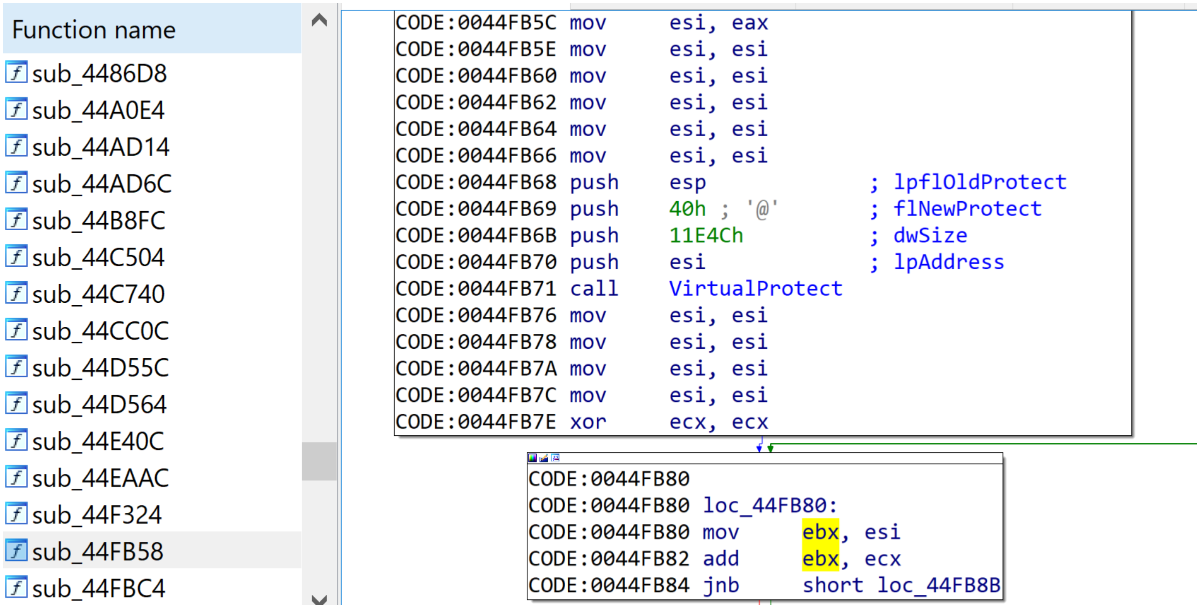
Difficult to interpret
The granularity is too small ☹

Easy to interpret ☺

Why Function-based Feature

- Easier to interpret than byte or basic block
- Often reused in a same malware family
  - Malware is rarely implemented from scratch
- Functions and their relationships are focused on during analysis

→ **Function Call Graph (FCG)**

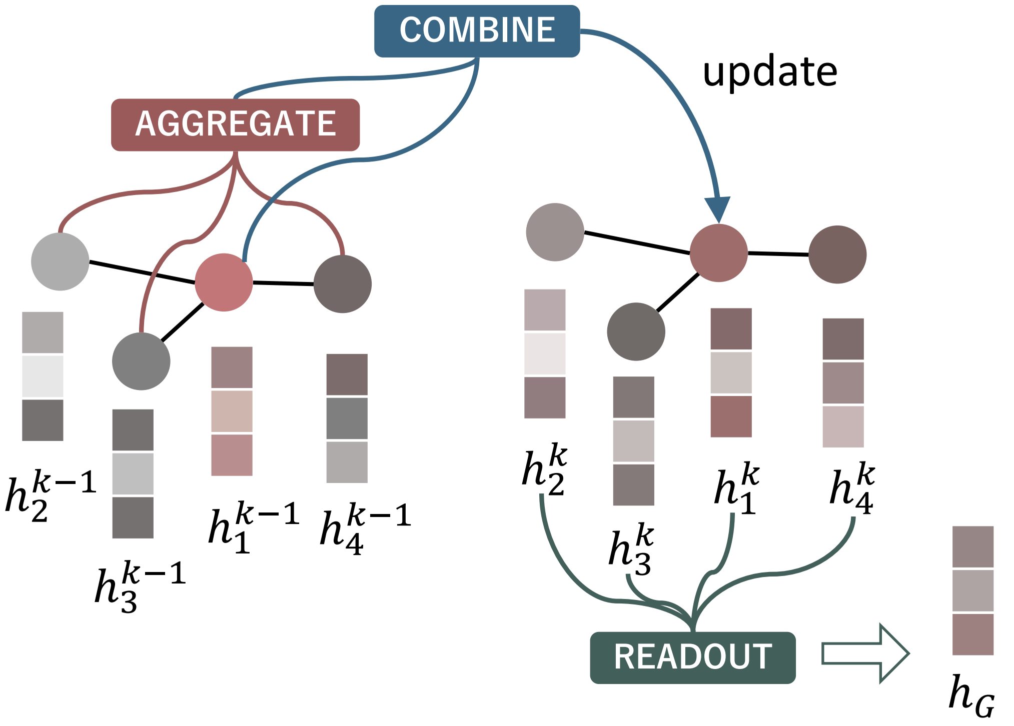


Graph Neural Network (GNN)

- Updating feature to reflect the graph structure

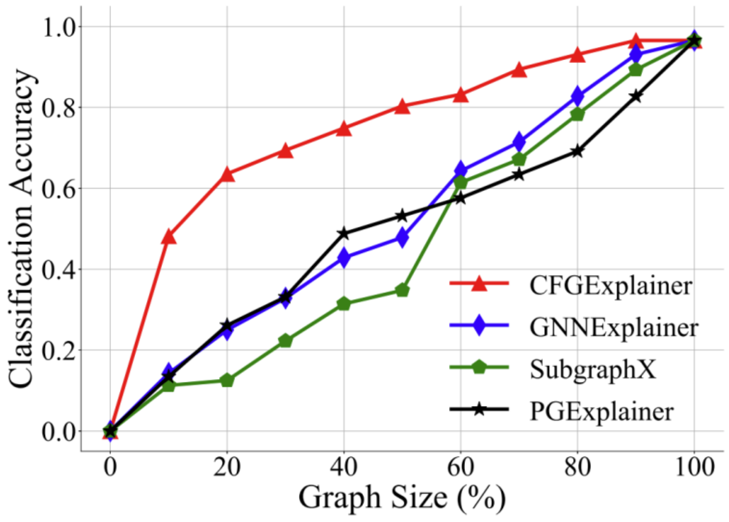
$$\mathbf{h}_v^k \leftarrow \mathrm{COMBINE}\left(\mathbf{h}_v^{k-1}, \mathrm{AGGREGATE}\left(\left\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\right\}\right)\right)$$

1. AGGREGATE:
   Aggregate features of neighboring nodes
2. COMBINE:
   Update the features of the node to be updated using the neighboring nodes
3. READOUT:
   Obtain a representation of the entire graph from the nodes in the graph



Related Work: CFGExplainer

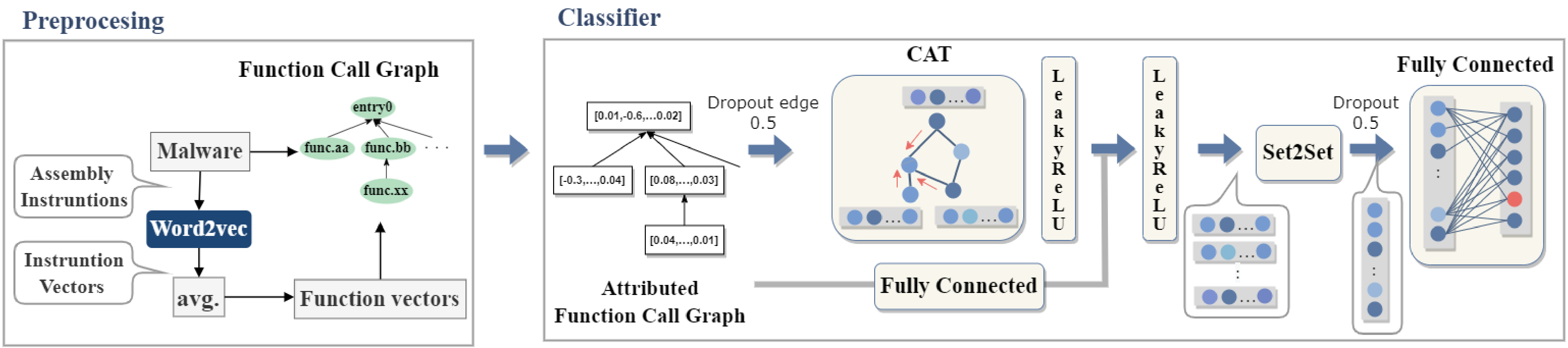
- Explanation method for GNN-based malware classification models
- Uses Control Flow Graph with Basic Blocks as nodes
- Identifies subgraph that contributes most to classification by pruning less important nodes
- Difference from our research:
  - CFGExplainer uses basic block, our FCGAT uses function



(d) Ldpinch

Herath et al. 2022. “CFGExplainer: Explaining Graph Neural Network-Based Malware Classification from Control Flow Graphs.” In 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE.

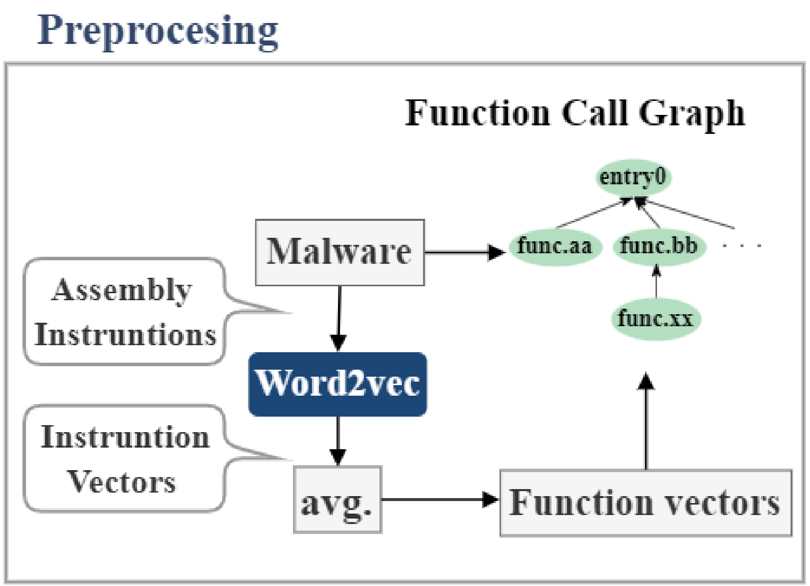
Overview of FCGAT



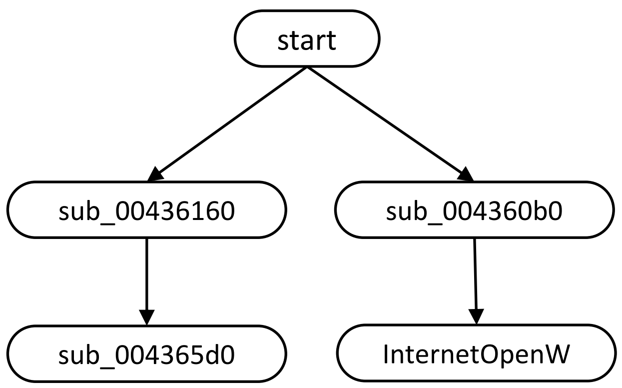
- Preprocessing
    - Creating Function Call Graph
    - Creating feature vector of function (function vector)
- Classifier
    - Malware classification using Graph Neural Network

Preprocessing

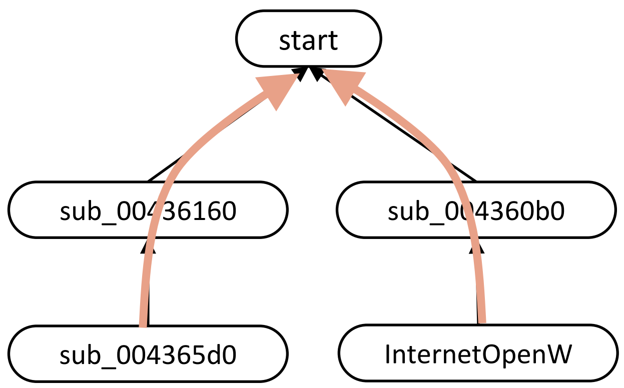
- Creating Function Call Graph
  - Using IDA Pro
  - Reversing arrows of FCG
    - The processing of called function is included in that of calling function



General FCG



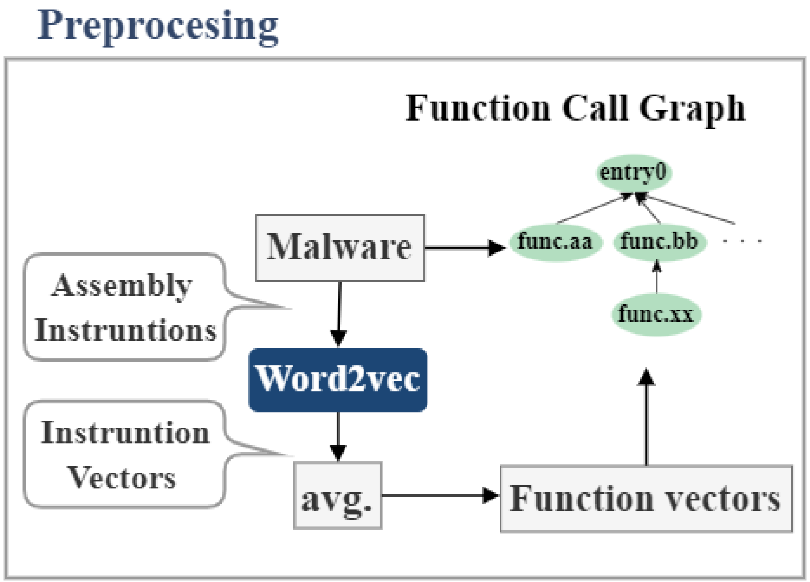
Reverse FCG



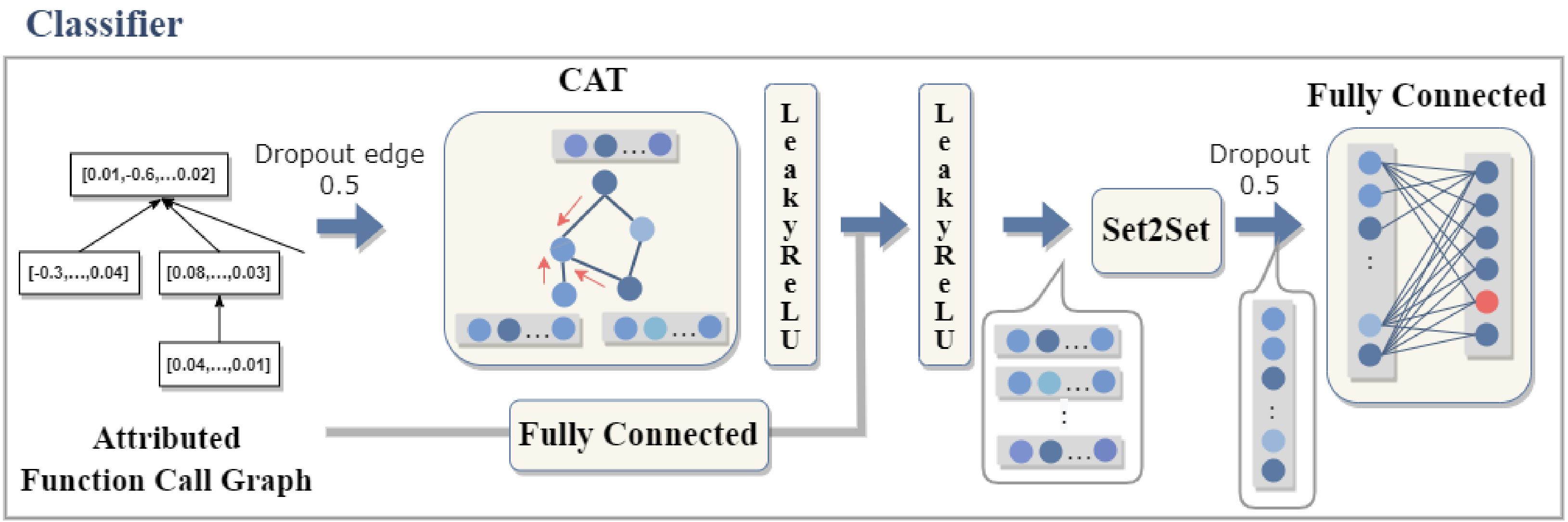
Preprocessing

- Creating Function Call Graph
  - Using IDA Pro
  - **Reversing arrow** of FCG

- Creating function vectors
  - Using **Word2vec** for creating instruction vectors
    - Instruction ↔ word,   function ↔ sentence
  - Averaging instruction vectors in a function to obtain a function vector



Classifier



- Graph Attention Network（GAT)
  - Updating function vectors using FCG
- Set2Set (readout process)
  - Next slide
- Fully Connected (final layer)
  - Classifying into a number of classes

$$\mathbf{u}_i = \mathrm{LeakyReLU}(\mathbf{W_1}\mathbf{x}_{v_i} + \mathrm{LeakyReLU}(\|_{k=1}^{K} \sum_{\mathbf{x}_j \in N(v_i)} \alpha_{ij}^k \mathbf{W}^k \mathbf{x}_j))$$

$$\alpha_{ij} = \mathrm{softmax}_j \left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top \cdot \left[\mathbf{W}_2\mathbf{x}_{v_i} \| \mathbf{W}_2\mathbf{x}_{v_j}\right]\right)\right)$$

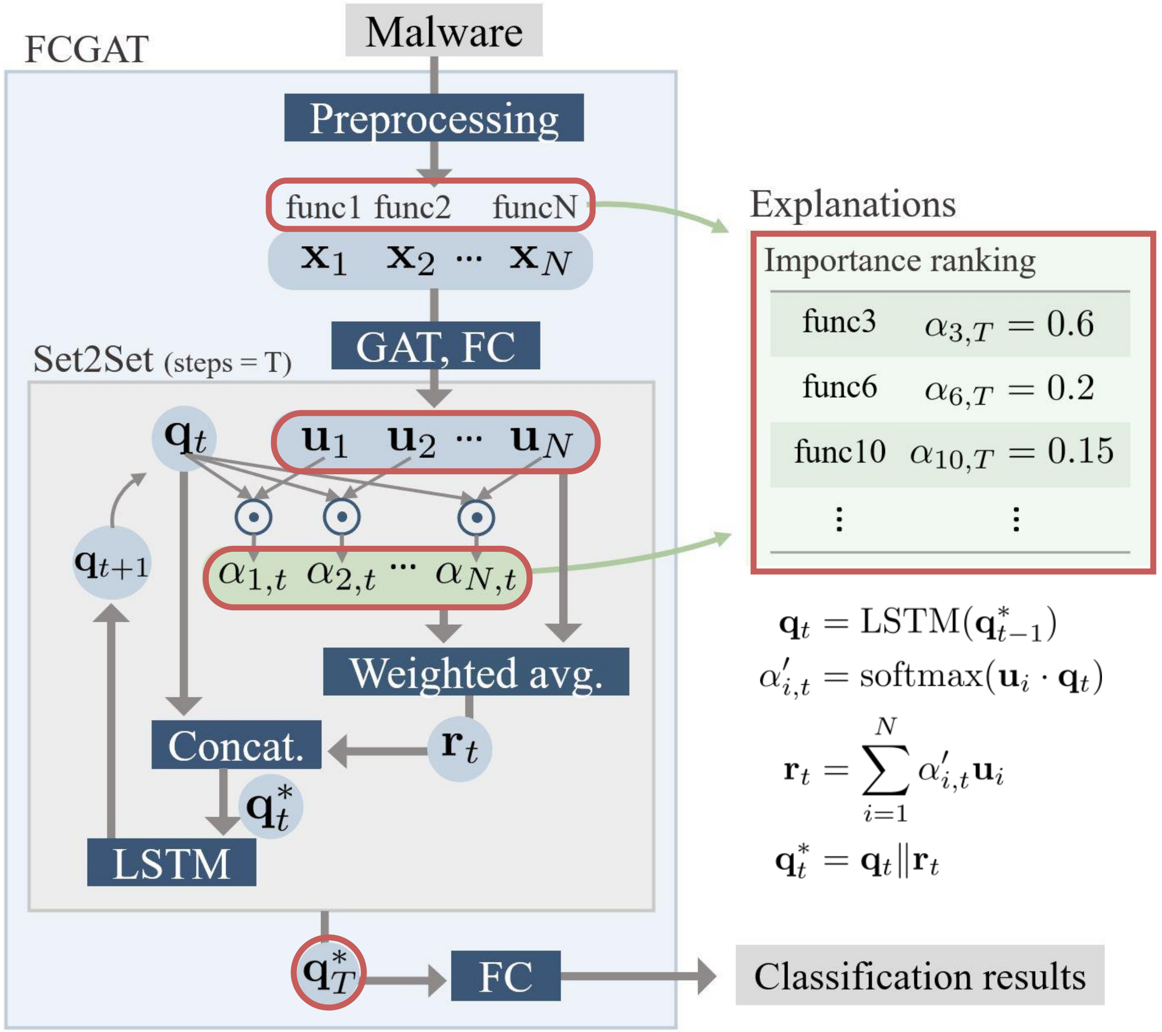
Set2Set (readout process)

- The key to interpretability

- Inputs : $\mathbf{u}$ Updated function feature
  Outputs: $\mathbf{q}_T^*$ Feature of the malware

- A larger $\alpha'_{i,t}$ (attention weight) is assigned to the more important function vector

- Importance ranking of function is provided



$$\mathbf{q}_t = \mathrm{LSTM}(\mathbf{q}_{t-1}^*)$$

$$\alpha'_{i,t} = \mathrm{softmax}(\mathbf{u}_i \cdot \mathbf{q}_t)$$

$$\mathbf{r}_t = \sum_{i=1}^{N} \alpha'_{i,t}\mathbf{u}_i$$

$$\mathbf{q}_t^* = \mathbf{q}_t \| \mathbf{r}_t$$

Experiments

- Classification **Performance**
  - Perform malware **family** classification
  - Compare with demonstration results of existing studies by Ma et al.
  - Conduct a replicated experiment of GEMAL (using FCG but not interpretable)

- Classification **Interpretability**
  - Perform malware **category** classification
  - Extract the importance ranking of functions as explanations
  - Confirm the effectiveness of these explanations

Ma, Yixuan et al. 2021. "A Comprehensive Study on Learning-Based PE Malware Family Classification Methods." In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 1314–25. ESEC/FSE 2021.
Wu, Xiao-Wang et al. 2022. "Embedding Vector Generation Based on Function Call Graph for Effective Malware Detection and Classification." Neural Computing & Applications.

Classification Performance

Category	Model	MalwareBazaar dataset				BIG-2015			
		Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Image	ResNet-50	96.68	96.91	96.75	96.83	98.42	96.57	95.68	96.08
	VGG-16	96.35	96.58	96.54	96.56	93.94	90.32	81.89	87.27
	Inception-V3	95.83	95.67	95.79	95.73	96.99	93.67	94.46	94.03
	IMCFN	97.38	97.53	97.41	97.47	97.77	95.93	94.81	95.13
Binary	CBOW+MLP	**97.81**	**97.92**	**98.08**	**98.00**	98.41	97.63	96.67	97.12
	MalConv	95.92	96.04	96.43	96.20	97.02	94.34	92.62	93.33
Disassembly	MAGIC	92.82	88.03	87.36	87.45	98.05	96.75	94.03	95.14
	Word2vec+KNN	95.64	93.34	94.29	93.79	98.07	96.41	96.51	96.45
	MCSC	96.80	94.97	94.51	94.70	97.94	95.97	96.17	96.06
	FCGAT	**98.11**	**98.03**	**98.27**	**98.15**	**99.27**	**97.93**	**98.45**	**98.18**
	GEMAL	97.71	97.65	98.00	97.82	**99.37**	**98.26**	**98.48**	**98.37**

by Ma et al.

by us

FCGAT outperforms all other methods on all metrics

FCGAT is equivalent to the replication experiment of GEMAL

Classification Interpretability

- Malware category classification
  - Malicious behavior is more common in categories
- Dataset: BODMAS-8cat
  - Exclude packed samples detected by peid from BODMAS

Category	Family Counts	Sample Counts
backdoor	31	598
downloader	19	967
dropper	17	397
informationstealer	19	347
ransomware	18	169
trojan	282	15,674
virus	8	93
worm	87	5,124
total	481	23,369

Yang, Limin et al. 2021. "BODMAS: An Open Dataset for Learning Based Temporal Analysis of PE Malware." In 2021 IEEE Security and Privacy Workshops (SPW), 78–84.

How much do *important* functions contribute to classification?

- Measuring the classification performances of subgraphs

Importance ranking

func3	$\alpha_{3,T} = 0.6$
func6	$\alpha_{6,T} = 0.2$
func10	$\alpha_{10,T} = 0.1$
func52	$\alpha_{52,T} = 0.01$
func30	$\alpha_{10,T} = 0.005$
⋮	⋮
func70	$\alpha_{10,T} = 0.0000001$
func33	$\alpha_{10,T} = 0.00000001$

1% subgraph

…

10% subgraph

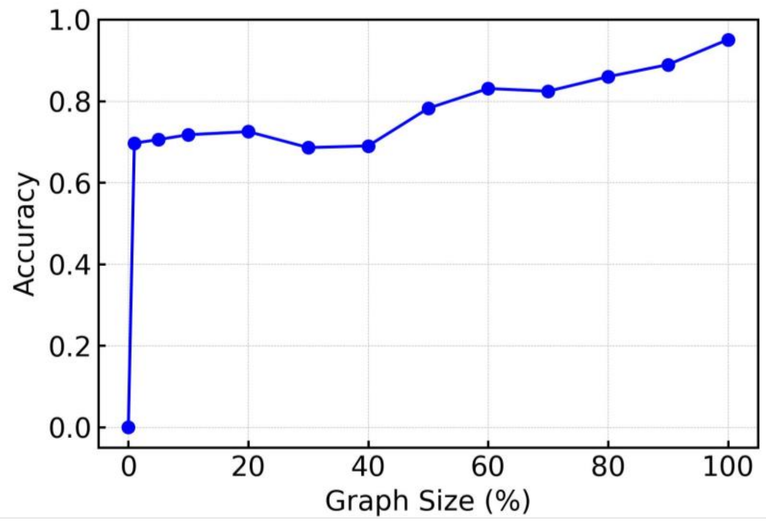
90% subgraph

100% subgraph = original graph

Evaluate classification performance

Classification Accuracies of Subgraphs

- Only top 6 functions (average per sample) achieve 69.67% accuracy
- Comparison with existing study
  - CFGExplainer showed 52.39% accuracy in the 10% subgraph
  - FCGAT achieves 71.73% accuracy !
- Using function, malware can be characterized with a small number of nodes



Graph Size (%)	Average Number of Nodes	Accuracy
1	6.2	69.67
5	28.8	70.53
10	56.9	71.73
20	113.4	72.48
30	170.1	68.57
40	226.5	68.99
50	283.0	78.20
60	339.6	83.06
70	396.2	82.40
80	452.7	85.96
90	509.3	88.92
100	565.4	95.06

Trend Analysis of Malware Categories

Aggregate results for each category of functions with max attention weights

backdoor		downloader		dropper		informationstealer	
CreateFileA	119	InternetOpenW	260	sub_460d8c2	278	GetFocus	202
fclose	105	MessageBoxA	130	GetWindowThreadProcessId	48	mciSendStringA	47
sub_4f847a1	83	mciSendStringA	77	CheckSumMappedFile	17	__dllonexit	11
SetWindowLongA	74	sub_f9da9b9	61	IIDFromString	14	free	10
GetCommandLineA	47	sub_1236153e	48	IsProcessorFeaturePresent	4	CreateFileA	9
sub_a9f1051	29	DispatchMessageA	34	sub_62922e7	3	sub_d0f95e9	8
sub_cf1fee5	18	UpdateWindow	29	__imp_VirtualProtect	2	lstrcatA	7
_fdopen	17	CoUninitialize	27	ShellExecuteA	2	GdipCreateFromHDC	7

ransomware		trojan		virus		worm	
CoRegisterMallocSpy	34	sub_a9f1051	1,316	GetActiveWindow	38	__abnormal_termination	1,314
CoReleaseMarshalData	22	__vbaUI1I4	1,036	strcpy	31	EnterCriticalSection	892
InternetReadFile	19	CloseHandle	709	start_80bc47b	15	sub_f7f9a83	407
ReadFile	12	InternetOpenW	633	?ProcessWndProcException	2	__ZNSt6locale5_ImplC2Ej	255
SetPropA	11	GetSystemDirectoryA	512	nullsub_3	2	__ZNSt6locale5_ImplC1Ej	205
SwitchDesktop	11	sub_acc2cf0	510	CoUninitialize	1	sub_d0f95e9	188
IsProcessorFeaturePresent	10	OleSetMenuDescriptor	472	GetFileVersionInfoSizeW	1	__ZNSt6locale6globalERKS_	182
CreateOleAdviseHolder	7	rtcLowerCaseVar	388	EnableMenuItem	1	SetWindowsHookExA	158

- Some functions reflect category characteristics

Case Study: Ransomware/ GandCrab

- GandCrab is ransomware that appeared in 2018
- We will see two samples, GandCrab#1 and GandCrab#2

DLL file | EXE file

GandCrab#1		GandCrab#2	
function	$\alpha'_{i,4}$	function	$\alpha'_{i,4}$
aes_encrypt	0.5984	aes_encrypt	0.0331
aes_decrypt	0.4015	aes_decrypt	0.0084
sub_10007BB0	2.002e-06	SetHandleInformation	0.0080
sub_10003F70	4.166e-07	GetTickCount	0.0080
sub_10004C20	7.423e-10	InitializeCriticalSection	0.0080

- These samples are common to the top two important functions, *aes_encrypt* and *aes_decrypt*, which are characteristic of ransomware

Conclusion

- Proposed FCGAT
  - The first study to explain malware classification on a per function basis
- Evaluated classification performance
  - High performance competitive to the latest method
- Confirmed the effectiveness of the explanations
  - Functions that reflect the malware feature
  - A small number of functions characterizing malware

Contact: mgs227501@iisec.ac.jp