# Automating Cybersecurity Attack Behavior Analysis and **Detection using Graph-driven Algorithms**

Yonghyun Kim, Madhukar Shrestha, Jeehyun Oh Faculty Advisors: Dr. Junghwan (John) Rhee, Dr. Grace Park, Dr. Fei Zuo

### PROBLEM AND MOTIVATION

According to American Advisor Groups, there is a cyberattack every 39 seconds, an average of 2244 incidents a day. These cyber-attacks involve exploiting software's weaknesses, common in most programs. Once attackers break into a computer system, in addition to the damage to the software often pursue privilege escalation and other following attacks with stronger Capabilities to pursue further and wider damages.

Cyber-attack damages may encompass files, accounts, software, and operating systems. Automating cybersecurity incident analysis and narrowing down the scope of analysis is essential to speed up the analysis and respond to the attacks faster, which is an important value for Security Operation Centers (SoCs).

### <sup>2</sup> OBJECTIVE

This project focuses on designing and implementing automated attack investigation and detection techniques based on graph structures.

- Due to a large body of information, we implement a backtracking mechanism to automate a laborious intrusion analysis.
- We implement an attack detection approach that compares a benign state and a state under an attack detecting suspicious activities based on divergence from the normal graph structure.

# **3 BACKTRACKING ALGORITHM**

foreach event E in log { /\* read events from latest to earliest \*/ foreach object O in graph {

- if (E affects O by the time threshold for object O) {
  - if (E's source object not already in graph) {
    - add E's source object to graph
    - set time threshold for E's source object to time of E
  - add edge from E's source object to E's sink object

The backtracking algorithm reconstructs a timeline of events during a cyber-attack by following their dependencies.

1. The backtracking algorithm goes over all the processes and saves timelines and information about them.

2. After reviewing all the processes once, it goes back to the beginning to analyze the relationship between each process. 3. If processes do not have any relationship with the suspicious process, it is filtered the unrelated process out.

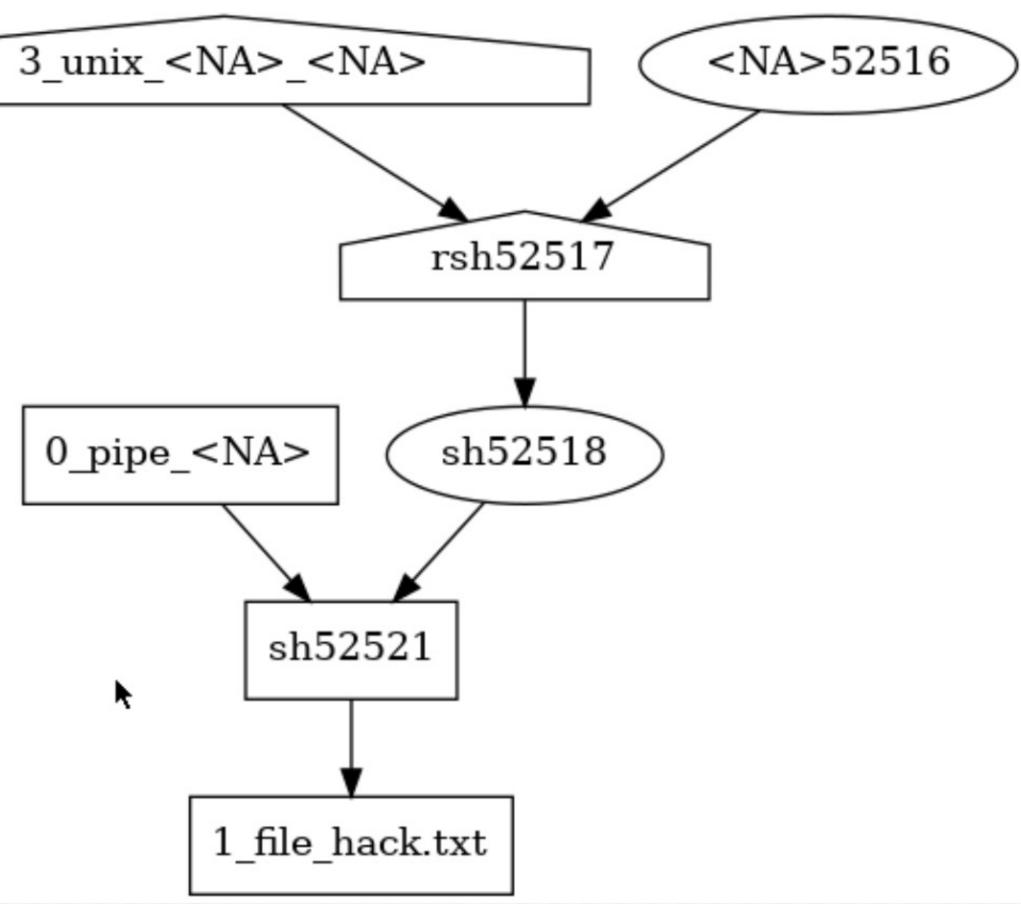




#### DEPENDENCY GRAPH

0 16,9,50	434,172.31.39.39	B.10.45885.4	Kka	13,19,459
connect?(190, 'connect')	('connext', 190)	180, 'ustaect')	Timmer', 100	0
- ipte	Ed malvel34			
fim	'weiler") ['weiler', 1921]			
14.jp=4_v=N	ir H.jpel, sBlir			
				_
			2,04,<84>	6 file cities.

### **REDUCING THE GRAPH**



- algorithm.
- activity is only remained.





(hesh49542)	contraineed shim/02422				14
Check, 5)	18, York')				
45390_172.31.39.30 16_kp_eNAs_eSMs	opache/252443 9.10=4	eNAs 10_jp_eSite_eNAs	10.jp.37662_127.8.8.1 10.jp.54776_127.0.0.1	10_jp_43651_127.0.0.1 10_jp_46766_127.0.0.1	13_saix_eN/e
(Vonnext', 183) (1246, 'connext')	(Seek, 20) [13, Seek] (153, York) [York, 15] [York, 11]	[ 20ad], 11] (17, York) (Year), 15)	['onnect', 1667'onnect', 167] (177, 'onnect')T'o	mascr. 1091TEM, 'unascr.)	(real, 35)
apashe212487	apashe212109 (apashe212122 (apashe212101)	apacite151999	apashe212580	0	navaged worked167
['span', 83] ['span', 82] ['span', 81] ['span', 82] ['bok', 83] ['slaw', 137]	(88, 'span') ['span', 153] ['span', 154]	(Tapen', 12) (Tapen', 13) (Write', 14) (18, Write')	(176, 'slow') [flock, 186]		
10, No. 14 on carbo 10, No. 10, No. 10, 20, 20, 20, 20, 20, 20, 20, 20, 20, 2	10.8a.heeks R.Me.groep R.Me.rgroeps.man (1.+NAr_+MAr	10_Ma_index php 7_file_access ing 1_vNike_vNike_vNike	E.jp.80,172.18.0.2 10.jpv4,vN8z epsche20212124		
(Tork, 94)			(3ek. 18	75	
aparho252517			Capache 252525		
Tchese, 951 T86, 'done' Tchese', 971 Tchese', 981 Tchese', 981 Tchese', 981 Tchese', 981 Tchese', 1981 (chese', 181) (chese', 182) (1	does', 183)(104, 'does')('does', 305)('does', 306)['d	see', 107)('does', 108)			
lar_shasta_accessing if_errentpail_childs 11_pape_childs 11_pape_childs 11_pape_childs 11_pape_childs 11_childs_childs 11_childs 11_childs 11_childs 11_childs_	NA>_CRA>_CRA>_CRA>_CRA>_CRA>_CRA>_CRA>_CR	0,030,000 21,030,030			

• The above graph is a part of the whole complex graph.

Each node indicates a process, a file, or a network address. Edges connect processes and the resources that are used. Therefore, we can see the process hierarchy and the detailed attack behavior occurred during an attack scenario.

• The total number of nodes of this big graph was 53880.

Despite the total number of dependency graph nodes is as high as 53880, the number of nodes essential to an attack can be reduced to 7 using the backtracking

Unrelated activities were filtered out and the suspicious

Due to the above simplified dependency graph, we can easily find the "1\_file\_hack.txt" file is resulted in the victim computer after an interaction of a few processes.

# 6 EVALUATION RESULT

Vulnerable Case	The number of nodes in the whole graph	The number of nodes in the reduced graph	% of the reduced graph
CVE-2018-19518	53880	7	0.000123
CVE-2021-41773	47968	10	0.000208
CVE-2021-42013	49720	10	0.000201
CVE-2021-35042	193549	4	0.000021

# 7 CONCLUSION

- effectively.

# **8 FUTURE WORK**

# **9 REFERENCES**

2018. 19, 2020

• The result shows the implementation of backtracking algorithm filters out nodes unrelated to the cyberattack

• There result shows that the algorithm can effectively make the analysis and detection of cybersecurity attacks more efficient.

• We are working on an automated intrusion detection algorithm using a deep learning technique and natural language processing.

 Combining graph reduction techniques and AI based attack detection mechanism can produce an efficient cybersecurity solution.

• Samuel K., Peter C., "Backtracking Intrusions", Department of Electrical Engineering and Computer Science at University of Michigan. 19 October, 2003. • Martin G., Martin R., Matthias K., Bogdan F., Erhard R. "Intrusion Detection on System Call Graphs. February,

• AAG. "How often cyber attacks occurs?". Web. June